CAAD

Publicación para Aventureros

FEBRERO-MARZO 2000
Año XI - 4ª Epoca - 12ª Suscripción
35

EL AÑO DE LA AVENTURA



I CONCURSO DE MINI-AVENTURAS

EL SELLO
HERNÁN CORTÉS
OPA-OPA
BALDUR'S GATE
SILVER
ZORK: GRAN INQUISIDOR
EL SULFATO ATÓMICO



LOS ROGUELIKE
CÓMO SE HIZO... SINTAC
GLK
CLASES PARA AVENTURAS

SUMARIO



CAAD Nª 35

(12ª Suscripción)

Publicación bimestral

(¡Esta vez de verdad!)

Ediciones CAAD

Apartado de correos 319

46080 - Valencia (Spain)

Telf.: (96) 369 95 71

E-Mail: caad@arrakis.es

http://www.cir.es/caad

Han participado

El director

Juan J. Muñoz Falcó

El Subdirector

José Luis Cebrián Pagüe

Los Redactores

Javier San José

Jaba

Manowar

Jaydee

Lumpi

Roger Amich

Daniel Cárdenas

F.J. Suñer

EDITORIAL

Por: Javier San José

En vuestras manos teneis un número del CAAD muy especial, un número fruto de la colaboración de varias personas que se han esforzado por poner lo mejor de ellos mismos en unas pocas líneas (a veces no tan pocas) que luego iban a ser publicadas en este, vuestro fanzine.

Este número es bastante especial ya que coincide con lo que podríamos denominar como el resurgimiento de la aventura conversacional en España. Varias actividades se han llevado a cabo en estos últimos meses. A nuestras espaldas queda Madridaventura'99, la más importante reunión de aventureros de todo España desde aquella mítica Barnaventura. Atrás queda, también, el primer Concurso de Mini-Aventuras en Castellano y ya deja paso a la segunda convocatoria del mismo.

Ha habido nuevas incorporaciones al "mundillo" lo cual demuestra que este género aún atrae a aficionados. Desde estas páginas quisiera dar una calurosa bienvenida a todos ellos.

Pero no debemos dormirnos en nuestros laureles. Debemos ir pensando en el futuro de nuestra afición. Esto es cosa de todos y todos debemos contribuir para que este género de juegos, que muchos llevamos cultivando desde hace años, no desaparezca. Es cierto que no es un género masivo y por eso es necesario que nos esforzemos más. Varias actividades se plantean para el futuro: nuevas convocatorias del concurso de Mini-Aventuras, una nueva "quedada" aventurera, por citar algunas.

Como digo, han pasado muchas cosas que demuestran que la afición española sigue ahí. Hemos pasado tiempos difíciles, hemos pasado una auténtica "Edad Oscura". Hagamos de esta que empieza una "Edad Dorada" de la aventura conversacional en castellano y por supuesto del CAAD.

NOTICIAS

Por: La Redacción

Tras demasiados días de discusiones y cambios más o menos menores a las bases, al fin acaba de nacer un nuevo concurso de aventuras. A diferencia de los anteriores, este concurso es mucho menos ambicioso: se trata de una competición de "Mini-aventuras", donde se exige que las aventuras participantes sean extremadamente pequeñas (normalmente una localidad y un puñado de objetos). El plazo es breve, como corresponde a un concurso de aventuras que pueden hacerse en una tarde. Pero lo mejor es que comprobéis vosotros mismos de qué va el concurso leyendo las bases que se publican más adelante en este mismo fanzine.

Noticias informitas: una nueva revisión de la librería de Inform, la 6/10. Esto aún no afecta al InformatE de Zak, salvo que este revise e introduzca los cambios (al parecer, soluciones de bugs sobre todo) a su librería.

Este es un año bastante abundante para la aventura al fin y al cabo. Luis David Arranz ha puesto a disposición de todos su nueva aventura conversacional, Las llaves del tiempo. Se trata de una aventura de texto, ambientada con algunos JPEG y una banda sonora MIDI, y cuyo argumento trata de viajes en el futuro.

Después de varias revisione, está diponible la versión 1.1 de Visual SINTAC que corrige numerors bugs sobre versiones anteriores e incorpora una completa librería de inicio. También ha sido renovado el tutorial, ahora completo y se han puesto ejemplos de aventura escritas con este sistema. Más información en http://personales.jet.es/jsjlogin/sintac

Nos comunica Zak que ya esta disponible su nueva aventura para InformatE!, Fotopía. Se trata de una traducción de la aventura de Adam Cadre: "Photopia". Podéis obtenerla en la siguiente dirección: ***http://www.geocities.com/TimesSquare/Fortress/9939***

Están disponibles en la web las ediciones de "The New Zork Times" en formato

PDF. Podéis encontrarlas en el siguiente enlace: ***http://gschmidl.cjb.net/nzt/***

If-Legends es un sitio web que es una verdadera máquina de búsqueda de interactive-fiction: ***http://www.iflegends.org***

Ya está disponible la demo de Estar Guars, de Papada Soft, en la siguiente dirección: ***http://members.xoom.com/EstarGuars/demo.zip***

Luis Felipe Morales (aka Manowar) acaba de poner de nuevo on-line su página, en la que incluye un resumen de aventuras presentadas a la minicomp, la podéis encontrar en ***http://pagina.de/LuisFelipe***

Las hordas informitas parecen estar renovándose, ya que desde allende los mares nos llega la noticia de que se está creando todo un entorno visual para este veterano lenguaje. Se trata de Visual Inform que aún se encuentra en fase de desarrollo pero en estado bastante avanzado. Eso sí, de momento sólo se podrá disfrutar sobre plataformas Windows. Más información en: ***http://www.iflibrary.org/vinform/***

UNPAWS, este extractor de fuentes de aventuras hechas con PAWs dispone ahora de nueva versión, en la que únicamente se han modificado ligeramente los fuentes para permitir su compilación con gpc además de Turbo Pascal, y así incluir como incluye, la versión compilada para linux además de la MS-DOS. Podéis obtenerlo en ***http://www.grupo-ronda.com/~caad/ficheros/utilidades/unpaws02.zip***

Plugh! es un entorno de desarrollo visual para TADS (el primo-hermano de Inform). Actualmente está en fase alfa pero ya se puede ver qué aspecto tiene en cualquiera de las siguientes URLs: ***http://plugh.tsx.org*** o ***http://redrival.com/plugh***. Parece que hay una fiebre por crear entornos visuales para programar aventuras conversacionales. ¿Por qué nadie habrá pensado antes en esto?

Aquelarre, la ópera prima de Pablo D'Amico se encuentra disponible en el área de ficheros de la lista del CAAD. Se basa en un módulo del juego de rol del mismo nombre y está programada en Inform. ***http://www.onelist.com/files/caad***

Bien, en esta página podéis encontrar las aventuras que participan en la primera minicomp, que finalmente se votarán por jurado popular segun las bases en la lista del caad: ***http://www.grupo-ronda.com/~caad/minicomp/aventuras/***

FHABLAOO es una nueva librería para InformatE que posibilita la creación de diálogos basados en menús, como siempre lo podéis encontrar en:

http://www.geocities.com/TimesSquare/Fortress/9939/informate

Luis David Arranz nos informa que ya hay disponible una version con gráficos de su aventura "La Guarida Valshar", aunque comenta, eso si, que sólo se podrá ver en tarjetas VESA compatibles, puesto que no incluye drivers para otras tarjetas. La podeis obtener aqui.

IznoguZ (InzoguZ by Zak: is Not Only Graphics Under Zmachine, ¡vaya nombrecito!) es un nuevo intérprete de máquina Z que está siendo desarrollado por Zak. Usa Glk por lo que soportará gráficos y sonidos en aquellas plataformas en que esto sea posible. Es un proyecto en desarrollo pero ya se pueden comprobar los avances visitando: ***http://www.geocities.com/spinf_2000/shot.jpg*** o ***http://www.geocities.com/spinf_2000/Xshot.jpg.*** Lo mejor es que se incluirá soporte en la librería InformatE para el manejo de gráficos y sonidos.

Javier Basilio (Jaba) está preparando la versión 5.2 del DAP que permitirá hasta 65535 mensajes lo cual suple una limitación histórica del DAP que hasta ahora sólo permitía 400. La versión no está aún disponible pero lo estará en breve cuando Jaba termine las fases de prueba y depuración.

Luis Felipe Morales (Manowar) ha abierto una web denominada la Mini Central (http://www.pagina.de/MiniCentral). De momento esta web está dedicada casi en exclusiva a ofrecer información sobre las mini-comps que se han celebrado. Información como capturas de pantalla de las mini-aventuras, comentarios, opiniones. Además podremos encontrar comentarios de otras aventuras no relacionadas con las mini-comps. Según el propio Manowar se trata de *"una web que pretende ofrecer contenido aventurero y que espero que con el paso del tiempo vaya creciendo poco a poco"*.

Resultados finales de la 1ª Minicomp. El fallo del jurado ha resuelto con bastante claridad que "La Sentencia", de Pedro A. Ala, es la mejor de las aventuras de la minicomp, siendo ganadora en tres categorías aparte de en la final. Mención de honor requieren "El Entierro" y "Renegados", vencedoras de las categorías de Brevedad y Originalidad respectivamente. Podéis ver los resultados finales en la siguiente dirección: ***http://www.grupo-ronda.com/~caad/minicomp/aventuras/fallo.html*** y más adelante en estas mismas páginas.

Ya está en movimiento la segunda Minicomp, podéis ver las bases en : ***http://www.grupo-ronda.com/~caad/minicomp2/minicomp2.html*** y más adelante en estas mismas páginas.

RESULTADOS I CONCURSO DE MINI-AVENTURAS

	Orig.	Calidad Texto	Interac.	Juego Limpio	Brevedad	Votos	TOTAL
PAEE	6,94	6,67	2,50	4,22	5,94	9	26,28
Un cuento chino	6,50	7,55	7,10	7,70	8,80	10	37,65
El Elegido	8,60	7,45	5,35	6,45	8,75	10	36,60
El Entierro	7,59	7,73	4,73	6,82	**9,18**	11	36,05
El Genio	8,30	6,25	5,10	4,70	8,10	10	32,45
La Sentencia	6,78	**9,11**	**8,56**	**8,94**	8,06	9	**41,44**
Renegados	**10,00**	7,40	7,60	3,00	6,60	5	34,60

Más Original



Renegados

Mejor Escrita



La Sentencia

Más Interactiva



La Sentencia

Más Breve



El Entierro

Más Jugable



La Sentencia

Mejor MiniAventura



La Sentencia

EL SELLO

Por: Javier San José

Hace años leí gran parte de las obras del escritor Lovecraft. Para los que no le conozcan, este personaje, de difícil carácter, creó todo un mundo de terror que giraba en torno a los mitos de Cthulhu, un terrible ser, un dios terrible, que permanecía sepultado bajo un sello que en caso de romperse lo liberaría provocando una catástrofe de dimensiones cósmicas. Vamos, que mejor no estar para verlo... Muchas criaturas de aspecto innenarrable (tanto como sus nombres) pueblan los mitos de Cthulhu: Sub-Niggurath, Yog-Shototh, los Profundos, el perro de Tíndalos, los Ángeles descarnados, sólo por citar algunas, todas ellas terribles y capaces de provocar la locura de aquel que ose toparse con ellas.



Casi todas las novelas que escribió Lovecraft giraban en torno a estos mitos. Muchas de ellas nos presentaban a algún personaje que, a pesar de su ignorancia de la existencia de estos terribles seres, al final acababa por percibir pequeñas revelaciones de esa realidad que rodeaba a la humanidad pero que quedaba oculta de los inocentes ojos de la mayoría. Normalmente el conocimiento de los Mitos desemboca en la locura o en algo peor.

Muchas aventuras conversacionales se han escrito inspiradas en la obra de Lovecraft, tanto dentro como fuera de nuestro país.

La que nos ocupa: EL SELLO, escrita por Fernando del Casar es una de ellas.

La mayor parte de los argumentos que giran en torno a la obra de Lovecraft se basan en ponernos en la piel de un personaje que, debido a misteriosos sucesos, se ve envuelto en una serie de indagaciones que, eventualmente, le llevarán hacia el conocimiento de la existencia de los Mitos.

El Sello no es una excepción. El autor nos mete en la piel de Victor Loy, cuyo tío, John Marcus, se encuentra sumergido en las investigaciones de unos extraños asesinatos. Tras ponernos en antecedentes, John Marcus se va de viaje a Arkham a investigar unas pistas, entre las que se encuentra un sello de extraña procedencia, y nos



deja a nosotros a cargo de indagar los sucesos que se están desa-rrollando.

La acción nos lleva desde Boston hasta Providence, ciudad donde se desarrolla la mayor parte de la trama. También realiza-remos una breve visita a Arkham, ciudad emblemática dentro de la obra de Lovecraft ya que en la biblioteca de su universidad se guarda una copia del Necronomicón escrito por el árabe loco Abdul Ahazred, obra que desvela gran parte de los misterios de los Mitos.

Poco a poco iremos realizando indagaciones que nos irán desvelando misterios relacionados, supuestamente, con la desaparición de nuestro tío.

El desarrollo está marcado por el transcurso de días. Tenemos un límite de tiempo para resolver los misterios que se nos plantean ya que al cabo del quinto día ocurrirá algo terrible que terminará irremisiblemente con nuestras investigaciones.

Hay varios PSIs con los que podemos conversar y a los que podemos interrogar. El sistema de conversación con los PSIs es algo atípico dentro de las aventuras conversacionales, y es que en vez de usar el comando DECIR A XXX "YYY" usaremos un sistema de frases preparadas que seleccionaremos a modo de menú. En cierta medida el sistema recuerda al de Monkey Island y productos similares de Lucas Arts.

La mayoría de PSIs son estáticos, es decir, no se mueven de su sitio. Quizá nuestro mayordomo (mejor dicho el de nuestro tío), Jenkins, sea el más elaborado ya que se desplaza por toda la casa, y por fuera de ella también, desplegando sus actividades cotidianas.

La distribución de la pantalla es la típica con los gráficos en la mitad superior y el texto en la inferior. En la franja intermedia se nos presentan las conexiones de la localidad y un reloj que marca el paso del tiempo. Es muy importante controlar el paso del tiempo ya que, como se ha mencionado, la aventura trans-curre contra reloj y debemos resolver el misterio en un plazo de tiempo. Además se supone que debemos comer a nuestras horas (Jenkins se encargará de servir las comidas a las horas concretas) y dormir cuando se haga de noche.



El inventario se muestra de forma gráfica, a pantalla completa. Está compuesto por seis casillas (el máximo número de objetos que podemos llevar simultáneamente) que contendrán los gráficos de los objetos que llevemos.

Un objeto casi imprescindible en el juego es el periódico. El periódico no permite estar al día de los sucesos que van ocurriendo a lo largo del juego y nos sirve para encaminar nuestras investigaciones en la dirección correcta. Jenkins, como siempre tan servicial, se encargará de comprar la prensa diaria.

El personaje tiene cuatro parámetros a controlar: hambre, mente, cansancio y salud.

Mente se refiere a nuestra salud mental. El mero conocimiento de los Mitos puede alterar nuestro estado mental hasta caer en la más profunda de las locuras. El valor de Mente nos da un indicador de lo cuerdos que estamos, por decirlo de alguna manera.

La Salud la debemos vigilar ya que indica nuestro estado físico. Si no comemos, ni dormimos la Salud disminuirá progresivamente.

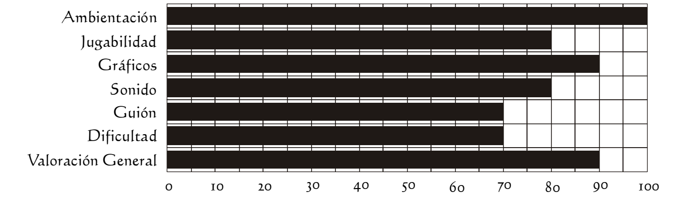
Hambre y Cansancio no necesitan explicación. Cuando alguno de estos parámetros tenga un valor demasiado alto deberemos comer o dormir, según corresponda.

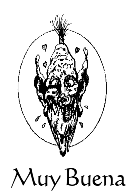
Los gráficos que incorpora la aventura están muy cuidados y ambientan perfectamente el juego. Es posible, incluso, observar como los personajes mueven la boca durante las conversaciones. Además la música que acompaña al juego combina perfectamente con la historia. Hay unas treinta melodías que se reproducen a lo largo del juego, en diversas situaciones, todas ellas se adaptan perfectamente al desarrollo de la aventura. Sergio Llata Peña, el autor de la música, ha hecho un trabajo excepcional (de hecho estoy escuchando alguna de ellas mientras escribo este comentario y suenan excepcionalmente bien).

Por cierto, la aventura está escrita para MS-DOS, usando el SINTAC aunque debería funcionar correctamente bajo Windows. Para escuchar el sonido necesitaremos una tarjeta de sonido compatible Sound Blaster. En definitiva, la mejor aventura conversacional española con la que he jugado desde hace mucho tiempo. Imperdonable dejarla pasar. Bien por NFC SOFTWARE y que sigan con su buen hacer.

Nota del Autor: deberéis perdonar si alguno de los nombres “lovecraftianos” no están correctamente escritos. Mi corrector ortográfico aún no incluye el diccionario de los Mitos y se empeña en subrayármelos en rojo. En fin. Qué le vamos a hacer... ■







LA BÚSQUEDA DEL OPA-OPA

Por: Lumpi

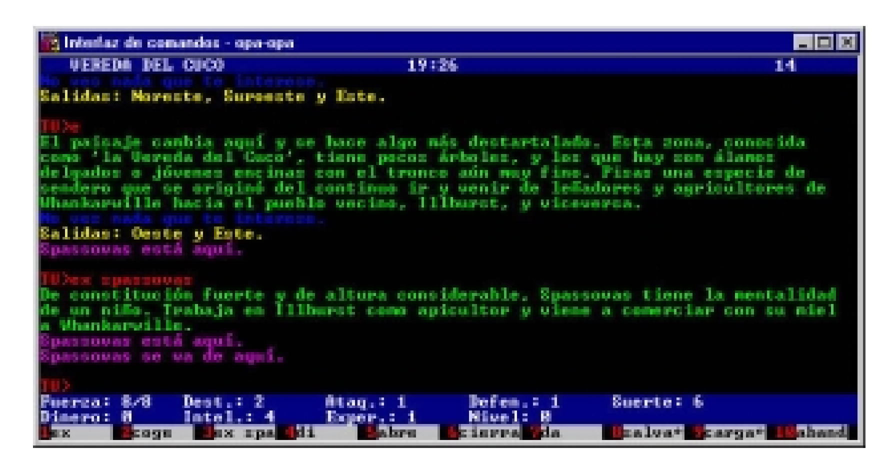


Lo primero que llama la atención, es sin duda el lenguaje utilizado por el autor para programar esta aventura, y es que se trata nada más y nada menos que del QBasic (si, el de msdos). Aunque parezca mentira, aún quedan valientes que se atreven a desarrollar aventuras conversacionales en este lenguaje. El resultado es sorprendente e incluso supera a muchas aventuras programadas en parsers especializados.

El juego incorpora casi todas las clásicas características de una aventura conversacional, pero también cuenta con algunos elementos que recuerdan bastante al RPG, tales como la posibilidad de luchar con otros PSIs o los parámetros asignados a nuestro personaje (fuerza, defensa, etc.).

La aventura, que comprimida sólo ocupa 81 kb (no tiene gráficos ni sonido), cuenta con un argumento algo simplón y típico; encarnas el papel del hijo de un leñador que de repente le ha dado por querer llegar a ser Caballero aventurero del Rey. Por supuesto, esto no se consigue así por las buenas. El primer problema será tratar de convencer al padre para que te "preste" algunas de sus armas, pero esto va a ser difícil, ya que al padre se le ha metido en la cabeza que su hijo debe seguir sus mismo pasos y convertirse en leñador. El segundo gran problema consistirá en llegar al castillo del rey y convertirse en Caballero (algo más

chunguillo que lo primero). No obstante, según palabras de su autor, este juego forma parte de una saga -la saga del opa-opa- por lo que para lograr acabar por completo este segundo objetivo habrá que esperar a la segunda parte (o a la tercera).



La presentación del juego consiste en una pantalla a modo 80x25 y con un tipo de letra bastante curiosa. En la barra superior izquierda se nos indica la localidad en la que nos encontramos. A la derecha podremos consultar el número de días que han pasado desde el inicio, y en el centro se aprecia la hora imaginaria del mundo creado, que irá avanzando a una velocidad de dos minutos por mandato. También tenemos en la barra inferior una serie de abreviaturas con algunas de las opciones más comunes, tales como examinar, salvar, etc... Un poquillo más arriba de esta barra se encuentran ubicados todos los valores de los datos que afectan al personaje (fuerza, destreza, ataque, defensa, suerte, dinero, inteligencia, experiencia y nivel). Finalmente está la parte principal y más importante de la pantalla, en la que se nos dará una descripción de la localidad actual, y si se da el caso, la lista de objetos, las salidas existentes y los nombres de los PSIs que se encuentren en dicha localidad. El resultado de todo esto es un entorno sencillo y bastante agradable de usar. Solo veo un problema, y es que tras el indicador (TU>) nos vemos obligados a escribir inmediatamente a continuación del signo ">". Se echa en falta un espacio que separe el indicador de la orden que estás escribiendo, aunque supongo que esto será cuestión de gustos.

Tras acostumbrarnos a este diseño y jugar un poquillo, nos damos cuenta del esfuerzo que el programador ha realizado en componer un parser agradable. Aunque en algún momento nos podemos encontrar con algún que otro odioso atasco tipo "acierta-verbo", el juego cuenta con un vocabulario más o menos extenso, e incluso existe un mandato capaz de mostrar en pantalla los sinónimos que acepta el juego de cada verbo u objeto, algo que siempre se agradece. Sin embargo existe un detalle que echo bastante de menos, y es que cuando se quiere leer de nuevo la descripción de cualquier localidad, ya no vale el clásico disminutivo "m" sino que tenemos que teclear "mirar" o "ex". Tenemos pues, que esperar unos cuantos minutillos más para intentar vencer la pereza que nos supone escribir más de una letra para consultar la descripción de una localidad...

Prestando atención al texto, nos daremos cuenta de que se aprecia en él una calidad

literaria medio aceptable. A lo largo del juego encontraremos muchas notas humorísticas, utilizando para acentuarlo un lenguaje coloquial, abundando mucho los vulgarismos.

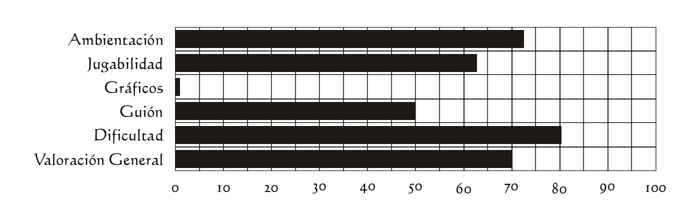
Los distintos PSIs que rondan por el juego, siguen en la misma línea de casi todos los que he visto anteriormente. Desgraciadamente he tenido que seguir escribiendo líneas y líneas siempre que he intentado sacar información de cualquier PSI. A pesar de ello, su comportamiento se podría calificar de "inteligente", ya que cada uno tiene asignada una función propia en el juego, y además poseen la capacidad de desplazarse por una ruta definida. Aún así, podemos dividir todos los PSIs en dos categorías: los civiles y los luchadores. Los civiles desempeñan la parte útil de la aventura. Con ellos podemos dialogar, entendernos, etc., algo que no pasa con los luchadores, con los que no se puede intercambiar ni una palabra. Al encontrarte con un PSI de esta índole, el juego cambia automáticamente a modo combate. Si no queremos morir, tenemos que luchar obligatoriamente, sin posibilidad de escape. Este aspecto podría haberse trabajado un poquito más, ya que las luchas son completamente no interactivas, y su resultado depende únicamente del azar, aunque claro está, lógicamente siempre influirá el valor de los parámetros de uno y otro contrincante.

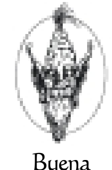
En cuanto a los puzzles, la verdad es que no nos encontraremos muchos a lo largo del juego. Los que hay son bastantes sencillitos para el iniciado, aunque puede complicarse para cualquier principiante. En general, la mayoría se resuelve interactuando con los PSIs de forma lógica. De todas formas este juego no es lo que se denomina una aventura lineal, ya que para pasarse el juego no hace falta descifrar puzzle a puzzle en un orden determinado, sino que el orden en que lo hagas no influye. En este aspecto, conviene destacar el factor tiempo, y es que como ya sabéis, el mundo en el que nos encontramos tiene su propio medidor temporal. Antes de las nueve, el color de la pantalla cambia para indicarnos que ya es de noche. Es importante tener este factor en cuenta, pues no es lo mismo una localidad de noche que de día.

Como ya he dicho antes, el juego carece de gráficos y sonido. A decir verdad, ninguno de estos dos elementos se echan en falta, ya que esta carencia se cubre con una aceptable ambientación, que a fin de cuentas es lo más importante.

Y con todo esto llegamos al final. Realmente esta aventura merece el adjetivo de recomendada, tanto para expertos como para novatos. Si os gustó "El poder de la kínbrenton", este juego es bastante parecido, aunque algo más simple y más corto. Solo queda esperar que la segunda parte sea aún mejor. ■

LA BUSQUEDA DEL OPA OPA





Buena

HERNAN CORTÉS

Por: El Caballero Heavy



l argumento de esta aventura es la aventura de la conquista del Nuevo Mundo por parte de Hernan Cortés.

En la primera parte debemos conseguir embarcarnos en una expedicion hacia Cuba (nos encontramos en Santo Domingo), para ello disponemos de tres días con sus noches.

En la segunda parte debemos conseguir embarcarnos desde Cuba para la conquista de Méjico.

La aventura contiene una introducción para las dos partes con música y gráficos tipo cómic (recuerdan mucho al tipo de dibujo de los cómics de Jabato o El Capitán Trueno) muy buenos. Este tipo de gráficos está presente también en todas las localidades de la aventura (todas las localidades tienen su gráfico).

La presencia en pantalla es realmente buena. Se ha usado alta resolución. El gráfico de localidad aparece a la izquierda, gráficos del inventario a la derecha y debajo de ambos hay un indicador de salidas, uno de tiempo simbolizado por la Luna o el Sol (depende de si es de noche o de día) desde que salen hasta el ocaso.

Los textos están bien y no veo faltas de ortografía. El comando examinar podemos (y debemos) usarlo a fondo ya que casi siempre hay una respuesta que no sea el clásico «no veo nada especial». También, a mi parecer, tendrían que haberse previsto más sinónimos (por ejemplo pescar con red), además de que la palabra «restos» de tabaco y «tabaco» no son lo mismo, ocasionando confusión al jugador en determinados momentos al usar una palabra por otra y ver que, por ejemplo, frente a un personaje causan reacciones completamente distintas.

Los gráficos de localidad cambian del día a la noche y no están sólo de adorno, ya que pueden ser examinados. Además la aventura está plagada de efectos sonoros, cuando abrimos una puerta, bebemos, etc (la aventura soporta unas cuantas tarjetas de sonido). También se han añadido animaciones, por ejemplo cuando abrimos/cerramos puertas o hablamos con un personaje este mueve la boca, y en otras escenas. En este sentido recuerda a la aventura Barbarian Quest, ya que los diálogos con los personajes se tratan de igual forma (una serie de opciones debajo el retrato del personaje).

También está muy bien el modo de tratar a nuestro personaje (Hernan Cortés, nosotros mismos). Se ha utilizado una serie de contadores (fuerza, salud, moral, etc...). Para determinar las condiciones en las que nos encontramos.

Estos contadores variarán según las acciones que emprendamos (por ejemplo dormir disminuye el contador de cansancio).

En cuanto al desarrollo de la aventura es entretenido, pero en algunos momentos quizás flojea un poco. También influye el hecho de que en el juego existen algunos «red herrings» (objetos inútiles) lo que dificulta, mediante la desorientación del jugador, el avance en la aventura.

En cuanto a los puntos negros, hay un par que he podido observar:

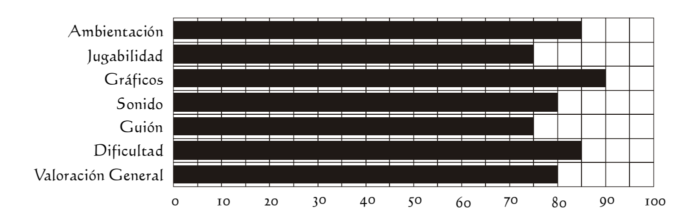
• Si nos atascamos en la aventura en la primera parte, me temo que el tiempo (los tres días) pasa demasiado deprisa para darnos tiempo a hallar una solución

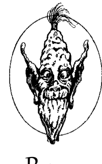
• El otro punto negro (muy grave y hallado en ambas partes) es que, aparte de algún que otro bug menor, tras cargar una partida grabada puede suceder que la aventura no nos permita coger objetos a pesar de que tengamos espacio para coger más, e incluso si dejamos nuestros objetos... ¡no podemos volver a cogerlos porque nos sigue saliendo el mensaje «No puedes cargar más cosas»! (también puede suceder que los mensajes al hablar con algún personaje se mezclen con otros y los gráficos del inventario también se líen).

Por lo que he podido ver estos bugs pueden aparecer tras cargar una partida grabada, por lo que lo más aconsejable si suceden es iniciar de nuevo la aventura y cargar de nuevo nuestra partida grabada, e incluso puede forzarnos a cargar una partida grabada más anterior que la última ... o a volver a empezar en caso extremo.

En definitiva, es una aventura entretenida, agradable de jugar y con buenos detalles, pero le falta depuración. ■

HERNAN CORTES




Buena

ZORK
GRAN INQUISIDOR

Por: Javier San José

Estamos en Puerto Foozle, pequeña localidad costera. Podría ser la típica aldea de pescadores, apacible y tranquila. El caso es que hace tiempo fue el lugar elegido por el poco brillante Mir Yannick, antiguo estudiante de la Universidad Mágica para fijar su residencia.

Debido a sus fracasos continuos en los estudios de magia, decide dedicarse a negocios más rentables como la religión. Pronto Mir Yannick asciende vertiginosamente en la jerarquía de la Inquisición.

La Inquisición es una organización que durante años ha perseguido la magia intentando erradicarla por completo. De hecho gracias a una extraña máquina, el Totemizador (de Electricidad Frobozz), consiguen aprisionar a todas las criaturas mágicas de Zork. Además se implanta la pena de totemización (algo muy muy malo) para aquel que haga uso de la magia en cualquiera de sus manifestaciones.





Nos encontramos en el 1067 GIS (GIS=Gran Imperio Subterráneo) y un nuevo proyecto de Mir Yannick, ahora Gran Inquisidor, está en marcha: la Inquisivisión (de Electricidad Frobozz), una máquina que desde la cima de Flathead Mesa pretende "ampliar" e "iluminar" las mentes de todos los ciudadanos del país.

Por supuesto nosotros llegamos a Puerto Foozle, ahora convertido en la base principal de la Inquisición, para acabar con los planes de Mir Yannick y devolver la magia a Zork. Para ello tenemos que recuperar una serie de objetos en los cuales ciertos magos han conseguido preservar la magia.

Contaremos con la ayuda de personajes tan curiosos como Antharia Jack (Dirk Benedict, más conocido como Templeton Peck "Fénix" del Equipo A) , protagonista de exitosas series (El Equipo Z) y películas (Great Underground Adventure III, de la que, curiosamente, no existen la parte I ni la II). También de Dalboz de Gurth, tercer Amo de las Mazmorras y antiguo compañero de Mir Yannick en la Universidad Mágica, y que nos acompañará durante toda la aventura ¡atrapado en un quinqué!.





Podremos visitar también a la excéntrica hechicera Y'Gael, viajaremos en el Metro Subterráneo, visitaremos la Presa de Contención número 3, la Casa Blanca (no la del presidente de Estados Unidos sino la casa al lado de la cual comienza la primera aventura de Zork). En fin, un delirante panorama digno de la saga Zork.

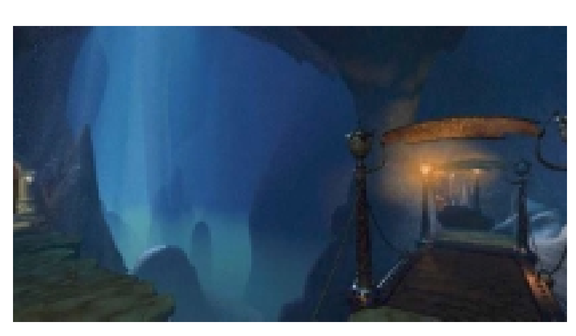
Nos encontramos ante una aventura gráfica con una excelente ambientación. Podremos ver el mundo de Zork en todo su esplendor y en primera persona ya que esta es la perspectiva que nos presenta el juego.

Los gráficos están compuestos en parte por escenas renderizadas, las cuales podremos ver en todas direcciones ya que el programa usa un curioso sistema denominado Z-vision que permite girar 360º para ver la escena, y en parte por secuencias de vídeo con actores reales.

Todo el juego presenta unas correctas locuciones en castellano (nada de textos o similares, bueno, algún letrero que otro nos encontraremos). Resulta realmente curioso oír a los actores hablando con acento portorriqueño. Todo este despliegue de medios se hace notar por los 2 CDs que ocupa el juego.

La interacción con el exterior, tanto con objetos como con personajes la realizaremos por medio del ratón. Podremos coger objetos, usarlos, lanzar hechizos, examinar los objetos y alguna acción más. El interface recuerda mucho al de las aventuras gráficas convencionales, usando el botón izquierdo del ratón para interactuar con el escenario y el derecho para acceder al inventario. Además hay un menú en la parte superior de la pantalla que se despliega al acercar el ratón hacia esa zona y que nos da un acceso rápido a los objetos que llevamos así como a los hechizos. Quizá resulte un poco molesto el sistema de examinar objetos que consiste en colocarlos sobre una zona del inventario, momento en el cual se ampliará la vista y nos mostrará el objeto con todos sus detalles (acompañado de una descripción del mismo de boca del propio Dalboz).

Los problemas que nos presenta el juego están bastante bien equilibrados y al principio nos encontramos los más fáciles, subiendo progresivamente la dificultad según avanza el juego. Hay que tener cuidado de examinar bien todos los rincones porque es bastante fácil dejarnos pistas importantes del juego al escapársenos algún detalle de las localidades que visitemos.



Además hay un punto del juego en el que pasaremos a controlar otros personajes para resolver alguna de las situaciones que se nos presentan. Aquí es donde quizá se podría haber ganado muchos puntos si la interacción con estos personajes se hubiese hecho más completa. Me explico. Estos personajes

aparecen totemizados y, en la situación correspondiente usaremos el tótem para "activar" el personaje, momento en el cual tomaremos control de él. Al final parece como si lo único que hubiese pasado es que se nos vacía el inventario, tomamos control de un cuerpo nuevo y poco más. Hubiese sido interesante poder interactuar con estos personajes de la forma tradicional (DECIR A DRAGON "VETE ARRIBA") pero para ello quizá hubiese hecho falta un interface más complejo que el que nos presenta el juego. De hecho no existen ni siquiera los típicos diálogos al estilo de los juegos de Lucas en los que un menú nos presenta las frases de la conversación. Una pena.



En su favor habría que decir que el juego permite un modo de multijugador cooperativo. Una curiosidad que no es habitual en juegos de este estilo. En este modo uno de los jugadores hace de guía y el otro de observador, pero pueden cooperar para resolver los problemas que se presentan. No es ninguna maravilla pero es curioso cuanto menos.

El juego está plagado de ese sentido del humor al que nos tiene acostumbrados la saga Zork. Realmente una de las cosas por la que merece la pena jugar a Zork: Gran Inquisidor es por ver las situaciones en las que se ve comprometido Antharia Jack, que parece tener una habilidad innata para meterse en problemas.

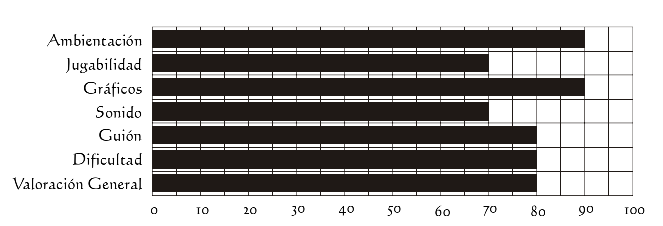
Si nunca has jugado a ningún juego de la saga Zork, no importa, este es, hasta la fecha, el único traducido a nuestro idioma y realmente merece la pena, aunque poco tiene que ver, en aspecto, con aquellas primeras aventuras conversacionales que iniciaron la saga. Además, el programa viene acompañado de una completa cronología de Zork para que nos pongamos en antecedentes.

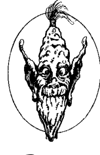
Si ya has jugado a algún juego de la saga este supone una interesante alternativa a aquellos cuya pantalla estaba llena de descripciones textuales y que se manejaban introduciendo órdenes por el teclado. Quizá algunos de los escenarios que tan brillantemente se nos presentan renderizados a pantalla completa te desilusione y no corresponda al aspecto de lo que habías imaginado leyendo las descripciones de texto de aquellas antiguas aventuras.

Por cierto, recuerda lo que dice Mr. Yannick Gran Inquisidor... **¿QUIÉN ES TU JEFE SUPREMO?. YO... YO SOY TU JEFE SUPREMO... YO SOY TU JEFE SUPREMO...** ■

ZORK: GRAN INQUISIDOR

Activision





Buena



MORTADELO Y FILEMÓN
EL SULFATO ATÓMICO

Por: Radical

Bueno ante todo decir que se nota el esfuerzo de un grupo de programación con entusiasmo, aunque las cosas a veces no salgan como uno quiere, como es Alcachofa Soft. (¿Quién es el que pone los nombres a las compañías españolas?, ¿Ehhhh Luis Felipe? X-D) .

Empecemos por ponernos en situación. Para los que no conozcan aún de qué va este juego (¿aún queda alguno que no haya leído el cómic?), El Sulfato atómico está basado en uno de los mejores cómics de estos personajes, en el cual se narraba como uno de los múltiples inventos del profesor bacterio (fallido por supuesto) cae en manos del dictador Bruteztrausen, el gobernante de la república de Tirania, y Mortadelo y Filemón son enviados a recuperarlo. Para ello tendrán que infiltrarse en el Palacio de Bruteztrausen, encontrar el sulfato y salir de allí IPSO FACTO.

El juego es sencillamente correcto, pero poco más se puede decir. Tiene una interface similar a la de su anterior aventura (Dráscula, para los olvidadizos) , la cual a mi gusto es engorrosa. Prefiero los típicos comandos en la parte baja de la pantalla al estilo Monkey Island (no confundir con The curse of Monkey island) aunque me tachen de purista.

Los gráficos y las animaciones sin duda son lo mejor del juego , al estilo de los cómics. Las voces, como cabría de esperar, no son las mismas de las de la serie de televisión, lo que decepcionará a algunos, pero que se comprende a tenor de lo que diré más adelante.

Los problemas a resolver son extremadamente sencillos en un principio y se complican de mala manera poco más adelante , lo que hace que muchos se cansen enseguida de él (es un coñazo que cada dos por tres te pille el guardia y tengas que volver a abrir la puerta, ¡y no te digo ya si es el otro personaje el que tiene las llaves!) .

Hay algunos gags que son realmente buenos, como cuando consiguen salir de la celda y Mortadelo canta una famosa canción de Nino Bravo que actualmente está muy de moda con el anuncio de una conocida compañía telefónica (¿aún hay alguien que no sepa de cual estoy hablando?) .

Y otra cosa es que es demasiado corto. Se nota que el final está hecho con prisas porque el salto al final es rapidísmo. No me lo podía creer. En un momento pasas del palacio, que es realmente donde se desarrolla todo el juego, al final. El juego es

caro para el tiempo de juego que ofrece.

Bueno, todo esto realmente estaría bien si Alcachofa Soft hubiera tenido todos los recursos disponibles para realizarla, pero no fue así. Me explico.

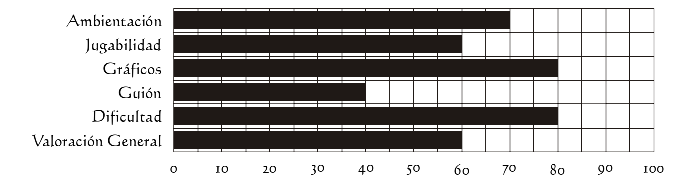
Según el propio Emilio, el juego realmente se hizo con prisas y el presupuesto pues no era todo lo bueno que uno puede desear. Pero la experiencia seguramente fue mejor que con la anterior compañía editora de DRASCULA (leáse DDM). Alcachofa Soft actualmente está en juicio con DDM por cambiar el código fuente original del juego (los detalles se explican en la entrevista realizada por Macedonia Magazine al propio Emilio). Desde aquí le deseo a Emilio toda la suerte del mundo.

EN GENERAL

Gráficos y animaciones muy buenas, puzzles que llegan a ser desesperantes en ocasiones, un sonido correcto y una interface algo molesta en ocasiones dan conjunto a este juego que se queda en lo que podía haber sido y no fué. ■

EL SULFATO ATOMICO

Alcachofa Soft

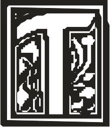




Regular

SILVER

Por: Daniel Cárdenas


ras nefastas experiencias con juegos como Ween The Prophecy y Dragon Lore, me había autoprometido no volver a desperdiciar el tiempo con más AGAs... Pero ahora, tras años de abstinencia, he de reconocer que he estado probando algunas de los más novedosas aventuras, empujado por la curiosidad de comprobar su evolución.

Sin duda, lo que más se lleva en estos momentos son los juegos del estilo del que nos ocupa, el de llevar un grupo de héroes a la rol con características, armas y tal, y con combates no por turnos sino en tiempo real. Las aventuras, nos guste o no, están derivando más a este género y la AGA clásica cada vez está más en vías de extinción (y si no os lo creéis, ahí está el ejemplo de Sierra, que tras Gabriel Knight 3 parece que cerrará el chiringuito en lo que AGAs se refiere).



Con respecto a esto, diré que estuve pensando incluso en renunciar a escribir este comentario, porque realmente este tipo de juegos no son aventura, ni siquiera AGAs, y lo único que hacen es ensuciar las páginas de este fanzine. Pero ya digo que en lo comercial la aventura deriva a esto, por lo que aunque sea un pecado comentarlo en el CAAD, sí se hace necesario.

LA HISTORIA

La historia es la típica de fantasía... El bueno, el personaje que controlaremos, se llama David y es el tipico héroe, muchachote guaperas, cachas y hábil con la espada. El emperador Silver, es el típico malo maloso, que ordena a sus sacuaces que le traigan a todas las mujeres, para él seleccionar una y casarse con ella. La mujer de nuestro muchachote le es arrebatada en una escena típica como ninguna. La misión, aunque después se complica un poco (sólo un poco), es la típica de encontrar unos objetos (ocho orbes mágicos) para derrotar al malo y así rescatar a tan dulce dama. ¿A qué os ha sonado típico? Es que lo es.

El comienzo sería disculpable si la cosa fuera mejorando a medida que se progresa en sus dos CDs, pero os advierto que no mejora, y que sigue siempre en la misma tónica. Además, el desarrollo es tan previsible que a veces hasta te ríes y todo (por no llorar).

INTERFACE

Sencillo y práctico al más puro estilo point&click, con la salvedad de usar la tecla CTRL en los combates (la publicidad miente vílmente al afirmar que la interfaz usa sólo el ratón). En estos, según como muevas el ratón tendrás estocadas, mandobles, paradas, etc. En cuanto a hablar con la gente y usar objetos, símplemente el puntero cambia avisándote al pasar sobre ellos, y la mayoría de las veces se activan sólos al llegar a la localidad adecuada. Para hablar ni siquiera se puede escoger lo que quieres decir de entre unas pocas frases, sino que todo está determinado y no hay opciones de conversación de ningún tipo.

Y tampoco puedes examinar nada... Para subsanar esto quizá te sirvan los excelentes gráficos pre-renderizados en alta resolución que son todo una joya: gráficos nevados, fortalezas, cavernas, bosques... Si, son fantásticos pero a mí personalmente no me sirvieron, y más cuando ves que cojea en otros quehaceres 'gráficos', como cuando al hablar los personajes y salir su cara en la esquina superior derecha, ésta se mantiene siempre estática. ¿Tanto costaba meter una animación o variar la cara cuando estuviera el personaje hablando, riendo o peleando? Fallar en algo tan básico le hace perder algunos puntos, porque... ¿qué son unos cuantos caretos en comparación con los doscientos gráficos de localidad que trae el juego?

CONCLUSIONES

En los anuncios de Silver, lo clasifican como 'juego de acción-aventura', pero hay que decir que esto no tiene nada ni de aventura, ni de rol. Silver es un arcade pasapantallas con pizcas de historia y un poco de estrategia en los combates, eso es todo. Es por esta razón que como aventura le pongo una puntuación tan mala. Si nos lo hubieran presentado sólo como arcade, lo habría considerado como tal y lo valoraría mucho más, pero ellos se lo buscaron al incluir la palabra aventura...



Lo que ya no entiendo es como se ha podido decir de este juego (en publicaciones supuestamente serias y especializadas) cosas como «fantástico argumento», «argumento repleto de fantasía», «todo un mundo de fantasías» y yo que sé que bobadas más. No sé, o no lo he visto todo (pero me la he acabado, ¡os lo prometo!), o los que dicen esas cosas no han jugado a una buena aventura ni leído una buena obra de fantasía en su vida o... o hay intereses económicos o de otro tipo por el medio.



En cuanto a los combates, el arcade en sí, hay que reconocer que están bastante conseguidos, con movimientos francamente espectaculares, el entrechocar de espadas y posibilidad de usar hechizos. Lástima que haya tantas y tantas peleas, porque al final acaban resultando pesadas.

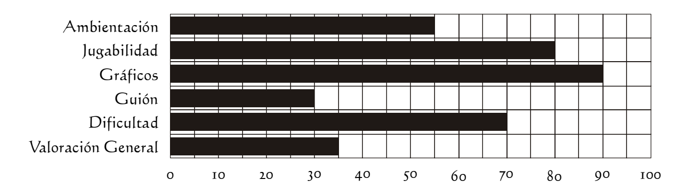
Los gráficos, sin duda, es el otro aspecto sobresaliente del juego, pero sólo con esto no se hace un buen juego. Desde luego Infogrames ha puesto mucho interés -y dinero- en ellos, con casi una veintena de grafistas, modeladores y dibujantes, pero han dejado de lado la historia. Y después, ni tan siquiera esta pobre historia está bien desarrollada, ni el mundo ni los personajes resultan creíbles y todo es muy lineal y está muy predeterminado. Vamos, que la historia y el desarrollo es de aficionados, y cualquiera de nosotros podría haberla trazado en menos de cinco minutos... Desde luego, a algo que cuesta 7000 pesetas y que resulta que es una superproducción resultado de varios meses de trabajo se le debe exigir muchísimo más.

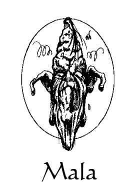
En serio, no se lo aconsejaría a ningún aventurero (si fuera gratis, aún). Aunque si lo que quieres es un arcade con el que ejercitar tus habilidades con el ratón, puede que te guste sino escarbas demasiado en él.

REQUERIMIENTOS

El juego es para Win9x, requiere DirectX 6 y un equipillo medio decente (mínimo Pentium 166 y 32 Mb). Lo más pesado son las interminables cargas al cambiar de pantalla o cargar situaciones, por lo que es muy recomendable una lectora de CDs rápida. Y de nada sirve, ir a la cocina por galletas o al lavabo a hacer un pis para ahorrarse la espera, porque lo más probable es que si la nueva localidad tiene enemigos presentes, que es lo más normal, al volver ya estés bien muerto... ■







BALDUR'S GATE

Por: Javier San José



etrás del sello AD&D (Advanced Dungeons & Dragons), correspondiente a un exitoso y ya veterano juego de rol de tablero, han surgido una multitud de productos de toda índole. Los más conocidos corresponden a módulos, enciclopedias de magia, enciclopedias de monstruos, figuritas de plomo, que complementan el ya mencionado juego de rol.



Además se han ido publicando varias series de novelas ambientadas en el universo que nos describe AD&D, un universo fantástico, lleno de magia y extraños seres. En nuestro país han sido publicadas ya varias trilogías y novelas independientes basadas en los Reinos Olvidados, un mundo construido en torno al sistema de AD&D. Gracias a estas novelas conocemos las peripecias de personajes tan emblemáticos como Drizz Do'Urden el elfo oscuro, Elminster el mago, Wulfgar el bárbaro.

En nuestras pantallas de ordenador ha habido varias series de juegos que reproducían más o menos fielmente el extenso universo de los Reinos Olvidados. Desde las antiguas de SSI (Curse of Azure Bonds, Treasures of Savage Frontier), pasando por las magistrales de Westwood Studios (Eye of the Beholder).



El juego que nos ocupa, Baldur's Gate, es uno más de los muchos que se han hecho sobre los Reinos Olvidades. Pero veremos lo que le diferencia claramente del resto.

La historia de fondo es la tan manida



de «malo tiene plan para conquistar mundo» y «bueno se lo impide». Eso sí, está perfectamente desarrollada. La trama se va entretejiendo a medida que avanzamos en el juego y vamos descubriendo cosas nuevas y desenmascarando los planes de nuestro adversario. Como muchos otros juegos de rol, empezaremos controlando un solo personaje. Posteriormente se nos unirán más personajes, hasta un máximo de seis, que nos ayudarán en nuestra gesta.

Los personajes están detallados, en cuanto a características y habilidades, según el sistema de juego de AD&D. Aquí hay que puntualizar que los programadores han conseguido plasmar el sistema de juego de AD&D de forma soberbia. De hecho, si alguna vez habéis jugado una partida de AD&D veréis en el juego reflejados todos los elementos característicos: varitas, pergaminos, habilidades especiales y todo el repertorio de hechizos, armas y armaduras que caracterizan a este género.

Un elemento importante en el desarrollo del juego y que, de cierta forma, influirá en los sucesos que se desarrollen, es el honor. Este atributo determina las reacciones de los personajes que nos encontremos hacia nosotros. Cuanto mayor valor tiene más es nuestra buena fama y mejor nos ven los otros personajes, cuanto menor es peor reputación será la que nos preceda y se nos cerrarán ciertas puertas. Es decir, en este juego podemos ir de «buenos chicos» o de «malos chicos». Es nuestra decisión el cómo comportarnos hacia los demás.

Hoy en día, con el «boom» de los gráficos 3D y las mega-tarjetas acelera-doras, todos esperamos que este sea un elemento «de serie» en cualquier juego actual. Baldur's Gate es la excepción que confirma la regla. Los gráficos del juego nos muestran una perspectiva isométrica del área. Nuestros personajes, meros «sprites», se mueven por estos escenarios. Los escenarios del juego, a diferencia de otros del estilo como Diablo, no están construidos por bloques, sino que se trata de escenarios totalmente renderizados que no muestran el aspecto monótono tan típico de los que están hechos a base de bloques. Esto, junto con la extensión de mapeado que nos presenta el juego, contribuye a que ocupe los 4 CDs que vienen en el paquete. El motor del juego muestra ciertos avances con respecto a lo que nos tienen acostumbrados los programadores. Podremos disfrutar de ciertos lujos como ver caer la lluvia, el resplandor de los relámpagos, efectos de ilumi-nación. Todo ello consigue crear un realismo no visto hasta ahora.





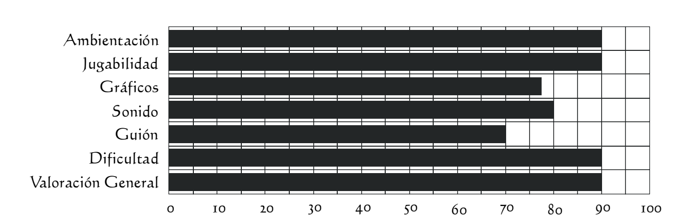
En cuanto a los combates se usa un sistema mezcla de turnos y tiempo real. Podemos combatir en tiempo real, dirigiendo las acciones de nuestro grupo pero en cualquier momento podemos parar el juego para elegir adversarios, seleccionar hechizos, activar defensas. El desarrollo de los combates se representa en pantalla en todo su esplendor. Veremos como nuestros guerreros asestan mandobles a diestro y siniestro, nuestros asesinos pueden maniobrar para dar puñaladas por la espalda y, por supuesto, nuestros magos harán todo su despliegue pirotécnico a 65000 colores. Hay que destacar que cada hechizo está perfectamente representado en el juego, tanto por los efectos que produce en los objetivos como por el despliegue gráfico que conlleva el lanzamiento de uno. Aquí hay que notar que, curiosamente, nuestros magos pueden llegar a memorizar hechizos de nivel superior al que nunca van a llegar ya que el juego limita el máximo número de puntos de experiencia que se pueden conseguir.

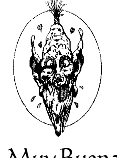
Es quizá este el único punto «negro» del juego y es que, prácticamente a mitad del mismo, nuestro grupo al completo ya ha conseguido la máxima cuota de puntos de experiencia con lo que, uno de los atractivos del juego, que es subir de nivel a nuestros personajes, queda eliminado. A partir de ahí ya sólo nos tendremos que conformar con seguir la historia para ver el desenlace final que, por cierto, es de lo más inesperado.

El sonido está bien conseguido, se acopla perfectamente a la acción y está al nivel de lo que se espera hoy en día de cualquier juego de este tipo. Vamos que no es como para «tirar cohetes» pero cumple su función. Además, en ciertos momentos del juego, hay escenas cinemáticas que también, sin ser una producción «hollywoodiense», cumplen con honra su cometido.

En resumen: un gran juego para los amantes de los juegos de rol al estilo AD&D, sobre todo para los seguidores de la serie de Reinos Olvidados ya que podrán encontrarse con ciertos personajes como Drizz Do'Urden y Elminster. Proporciona largas horas de diversión, es un juego que aprovecha al máximo las cerca de 3 Gb que ocupa. ■

BALDUR'S GATE





Muy Buena

CONCURSO DE MINI-AVENTURAS. 2ª CONVOCATORIA

El concurso tiene como objetivo fomentar la creación de aventuras, así como explotar las posibilidades del género en terrenos como la interacción con objetos y personajes. Además, servirá para ampliar la base de aventuras con código fuente disponible, como fuente de ejemplos para los distintos parsers y lenguajes de programación.

Para ello se sugiere que todas las aventuras participantes sean excepcionalmente breves y desarrolladas dentro de unas estrictas limitaciones. Ello permite desarrollar más concienzudamente aspectos como la interactividad y el detalle de objetos y escenarios. También ayudará a participar a aventureros que no disponen del tiempo necesario para hacer realidad una aventura completa y extensa.

Plazos

El plazo de recepción de los trabajos finaliza el 31 de Marzo del 2000 a las 12 de la noche.

Entrega

Las aventuras deben enviarse al CAAD por e-mail, a la dirección caad@arrakis.es. El envío deberá estar dividido en dos ficheros.

Un fichero contendrá sólo el ejecutable de la aventura. Si lo desea, el autor puede incluir un LEEME donde explique brevemente el objetivo de la aventura o cualquier cosa necesaria para jugarla. Este fichero será expuesto públicamente en cuanto finalice el período de recepción de trabajos.

Un segundo fichero contendrá un fichero de texto con la solución de la aventura y opcionalmente, el código fuente del programa. El autor puede incluir aquí cualquier comentario o información adicional sobre el juego, según crea él oportuno. Este fichero será expuesto públicamente cuando finalice el período de votación y con ello el concurso en sí.

La inclusión del código fuente de la aventura es opcional, pero muy aconsejable, ya que las aventuras que participen en este concurso serán ejemplos interesantes del uso de sus respectivos parsers y lenguajes.

Los trabajos presentados deben ser originales y no haber sido distribuídos antes del 1 de Enero del 2000. No se exige, sin embargo, que el juego quede oculto hasta el 31 de Marzo del 2000. El autor es libre de distribuirlo por su cuenta, a riesgo de mostrar su trabajo antes de tiempo (probablemente quedando en desventaja respecto al resto de participaciones).

Jurado popular

El día 31 de Marzo se habilitará una zona en la página del CAAD donde podrán descargarse los trabajos presentados. Un formulario en esa página permitirá realizar

votaciones a cualquiera que disponga de una dirección de e-mail.

Cada votación consistirá en puntuar (en una escala de 0 a 10) una de las aventuras participantes en el concurso en los siguientes apartados:

ORIGINALIDAD: ¿Nunca se había visto antes este problema o situación?

CALIDAD DEL TEXTO: ¿Está bien escrito?. ¿Carece de faltas de ortografía?.

INTERACTIVIDAD: ¿Ofrece el juego respuesta a la mayoría de comandos lógicos?. ¿Pueden manipularse a fondo todos los objetos?. ¿Dan los personajes bastantes respuestas?.

JUEGO LIMPIO: ¿Es un problema lógico?. ¿Hay pistas adecuadas si el problema es difícil?. ¿Se contemplan los intentos de probar otras soluciones lógicas al problema?.

BREVEDAD: ¿Es la aventura realmente una mini-aventura, o incluye un exceso de contenido?. Véase el siguiente apartado.

Una misma persona puede votar a una aventura, a varias, o a todas, sin ninguna obligación. Para que una aventura pueda optar a premio, debe haber recibido votaciones de al menos TRES personas.

Para realizar una votación, será imprescindible facilitar una cuenta de correo y verificar el voto respondiendo desde esa cuenta de correo a un e-mail automático. Esta pequeña molestia evitará votos múltiples y otras incomodidades. La lista de e-mails será destruída en cuanto finalice el plazo del concurso, para preservar el derecho a la privacidad de los votantes.

El período de votación finalizará el 30 de Abril del 2000. Se podrá ampliar el plazo si hay petición de ello y se observa que el número de votos recibidos no es suficiente.

Licencia

Ediciones CAAD no se reserva ningún derecho sobre los programas presentados al concurso. Éstos podrán distribuirse según el método y la licencia de uso que prefiera su autor, sin embargo todas las aventuras presentadas deberán poder distribuirse y ser jugadas gratuítamente, sin ninguna limitación.Todas las aventuras que incluyan el código fuente (lo cual recordemos que es opcional) deberán poder ser estudiadas, modificadas y reutilizadas libremente, por ejemplo utilizando parte de su código en otras aventuras.

Limitaciones

Dado que resulta extraordinariamente difícil determinar qué es una "mini-aventura", los requisitos de este apartado deben considerarse como sugerencias orientativas.

Ninguna aventura se rechazará por incumplir uno de los siguientes apartados, pero podría recibir una cierta penalización, ya que la brevedad es uno de los apartados a votar. Una aventura se considerará "breve" (10) si cumple a rajatabla todos los siguientes apartados:

- Incluir una sóla localidad, sin posibilidad de movimiento o al menos sin movimiento "auténtico", entre lugares distintos.

- Incluir un máximo de tres objetos o personajes (o cualquier otra combinación, como un personaje y dos objetos).

- Poder acabarse razonablemente en menos de 10 minutos, suponiendo que el jugador no queda trabado absurdamente en un atasco.

Se sugiere no penalizar una aventura en la votación de este apartado simplemente porque contiene objetos "de escenario", cosas examinables, o repuesta a muchas acciones. Estos aspectos enriquecen el juego sin alargar ni engordar su desarrollo.

Queda a discreción de la persona que vota qué es un objeto y qué no, y si el abuso de cosas como hechizos, objetos transformables u otros "pseudo-objetos" reduce el carácter de "mini-aventura" del programa.

Requisitos

La aventura podrá funcionar en cualquier plataforma, aunque una aventura recibirá más votos si todo el mundo puede jugarla.

La aventura podrá incluir atractivos adicionales, como música, gráficos, tipos de letra... Si bien se advierte que ello no se reflejará en la puntuación de la aventura.

Premio

No existe ningún premio para el concurso, salvo la satisfacción y el reconocimiento de todos los aventureros. Sin embargo habrá un ganador del concurso, elegido a partir de la media de todas las puntuaciones en todos los apartados.

El CAAD admitirá, sin embargo, cualquier donación de premio simbólico que alguien desee ofrecer al ganador, por lo que existe la posibilidad de que esto cambie posteriormente. Se elegirá un ganador del concurso, así como ganadores por apartados. En general, una vez finalizado el plazo de votación, se escogerán

LA MEJOR MINI-AVENTURA
LA MINI-AVENTURA MÁS ORIGINAL
LA MINI-AVENTURA MEJOR ESCRITA
LA MINI-AVENTURA MÁS INTERACTIVA
LA MINI-AVENTURA MÁS JUGABLE

En caso de empate, se organizará una votación de desempate, posteriormente al plazo de finalización del concurso.

Recordamos que cualquier mini-aventura que no reciba al menos las votaciones de TRES personas, quedará fuera de concurso.

CÓMO SE HIZO... SINTAC

Por: Javier San José

Hace mucho tiempo, en una galaxia... ehemm... perdón. Empecemos otra vez. Hace mucho tiempo, bueno, quizá no tanto pero sí hace unos cuantos años. Digamos que estábamos en 1992. Hacía poco se acababa de celebrar el primer Concurso de Aventuras. En dicho concurso quedaron siete aventuras finalistas. El primer premio consistía en un viaje a Valencia a conocer la compañía pionera en aventuras en España por aquel entonces, AD. Un viaje a Inglaterra para conocer la cuna de la aventura y, sobre todo, una sustanciosa cantidad de dinero.

El caso es que se nos propuso a los finalistas un trato. O se seleccionaba la aventura ganadora de entre las finalistas y se llevaba todo el premio, o renunciábamos directamente al premio y todos los finalistas recibiríamos una copia del DAAD.

DAAD=Diseñador de Aventuras de AD, era el programa (o «parser» como se suelen denominar) que los chicos de AD usaban para crear sus aventuras. Era el sueño de todo programador de aventuras de aquellos tiempos, de todos los que usábamos el PAW en nuestros exiguos Spectrums de 48K. La mayoría no lo dudamos mucho y aceptamos encantados el cambio de la posibilidad de ganar el premio por la certeza de obtener el DAAD

Al poco recibimos en nuestros buzones un par de disquetes de 5 ¼ que contenían el DAAD junto con la documentación. El DAAD necesitaba un PC para funcionar pero a cambio prometía un sinfín de posibilidades y facilidades que eran inimaginables en el PAW.

Yo no poseía por aquel entonces un PC en casa pero sí tenía acceso a algunos PCs. Rápidamente corrí a instalar el programa en un PC. Me decepcionó. El primer fichero de documentación que leí rezaba algo así como que la versión que tenía era una versión recortada. Resulta que el DAAD necesitaba un buen puñado de utilidades sobre todo para la creación de los gráficos y para «portar» las aventuras creadas a otros sistemas que no fuesen PC (Spectrum, Amstrad, Amiga, ...). Además para ejecutar algunas de esas utilidades se necesitaban los ordenadores mencionados.

Seguí leyendo la documentación. Me encantaron algunas de las características del DAAD como el concepto de indirección (los que hayan manejado PAW sabrán que los parámetros de los condactos eran valores fijos en cambio en DAAD se podía usar el valor de una variable para referenciar a otra).

Por aquel entonces yo andaba aprendiendo C, un lenguaje que me cautivó desde el principio por su potencia y versatilidad. Enseguida una idea se formó en mi cerebro (bueno, al menos en lo poco que me queda). Pensé que sería capaz de hacer un «parser» para PC

que fuese tan bueno como el DAAD, incluso algo mejor.

Empecé a programar la primera versión de SINTAC enseguida y para Agosto del 92 ya tenía lista la versión T1 (T de texto ya que era una versión que no soportaba gráficos, todavía...). El primer SINTAC ya incorporaba algunas características que lo hacían netamente superior al DAAD como la indirección en todos los parámetros (el DAAD sólo soportaba indirección en un parámetro) y los saltos incondicionales. Tamaño del código fuente 271 Kb.

En Enero del 93 ya estaba lista la versión T2 (todavía en modo texto) que incorporaba algunas mejoras y correcciones (condacto EXTERN, mejor gestión de memoria en el compilador, mejoras en el condacto INPUT). Una importante mejora fue la inclusión de la posibilidad de redefinir el tipo de letra y una utilidad que mostraba gráficos usando el condacto EXTERN (se conmutaba a modo gráfico, se mostraba el gráfico llamando a esa utilidad externa y se volvía a modo texto). Tamaño del código fuente 382 Kb.

El gran salto cualitativo llegó con la versión G1 (G de gráficos, por supuesto) hacia Abril del 93. Dicha versión ya incorporaba dos modos gráficos, uno de 640x480 y 16 colores y otro de 320x200 de 256 colores. Pero lo más importante de esta versión era que incorporaba un entorno de desarrollo integrado. Este entorno permitía editar, compilar y probar la aventura. Esto era un avance ya que facilitaba mucho el uso y aprendizaje del SINTAC. Tamaño del código fuente 762 Kb.

Por mi parte siempre he sostenido que cualquier entorno de desarrollo es mucho más amigable si viene acompañado de un entorno que facilite su uso. SINTAC permitía usar el propio entorno o usar un editor externo y compilar y ejecutar la aventura desde la línea de comandos. Por eso nació el EDS (Entorno de Desarrollo de SINTAC). Además disponía de un rudimentario, pero eficaz, sistema de ayuda en línea.

La versión G2, disponible hacia Enero del 94, mejoraba en cuanto a la cantidad de mensajes que había disponibles, ahora era posible tener hasta 65536 mensajes (256 tablas de 256 cada una) frente a los 256 disponibles hasta la versión G1. Además se incluyó un «linkador» para generar un ejecutable independiente lo que hacía más fácil la distribución de la aventura. También hubo cambios en cuanto al compilador de C que utilizaba. Hasta esta versión andaba usando el Quick C de Microsoft. En esta versión cambié al Borland C debido a que el proyecto empezaba a requerir una potente herramienta de desarrollo y por entonces la de Borland era la mejor (en mi opinión). Tamaño del código fuente 1 Mb.

La siguiente versión, la G3, estuvo disponible casi un año después aunque posteriores mejoras hicieron que la versión estable no estuviese lista hasta mediados del 95. Las mejoras estaban orientadas a dotar de soporte de ratón a las aventuras y de sonido a través del altavoz del PC (por aquel entonces ya tenía un PC en casa pero sin tarjeta de sonido). Esta ha sido la última versión del SINTAC. Realmente la versión G3 ha sido revisada en Agosto del 99 para corregir un fallo que llevaba presente desde la última versión que salió por el 95. Tamaño del código fuente 1,3 Mb.

En el periodo de 3 años transcurrido desde el primer SINTAC hasta el último debo decir que obtuve muchas satisfacciones que aún hoy en día se siguen manifestando en forma de aventuras creadas con este parser.

Debo dar mi agradecimiento a todos aquellos que depositaron su confianza en SINTAC, tanto los que se acercaron por curiosidad a él como a aquellos que llegaron a materializar sus aventuras con esta herramienta.

En el 98 comencé otro proyecto que a finales del 99 estará completado. Se trata de Visual SINTAC, pero... eso es otra historia. ■

CÓMO SE HIZO... VISUAL SINTAC

Por: Javier San José



odo empezó a principios del 98. Mi PC ya era lo suficientemente potente como para ejecutar Windows con soltura. Después de una temporada alejado del panorama aventurero me volví a acercar a él.

Lo primero de que me percaté fue de lo poco que se había avanzado en cuanto a las herramientas disponibles para desarrollar aventuras. Había habido un amago de proyecto de «parser» que tenía muy buena pinta y que se había denominado CAOS. Una página Web, ya abandonada, manifestaba los progresos que se iban realizando en este «parser». Realmente las características del este parser eran prodigiosas pero inexplicablemente el proyecto se vino abajo.

En mi cabeza hacía tiempo que rondaba la idea de portar el SINTAC a Windows. Hice algunos intentos pero fracasaron debido a la dependencia que había con el sistema operativo DOS. Así que me decidí a reprogramar todo el sistema desde cero.

Me marqué unos objetivos claros.

Primero, el sistema debía ser visual, es decir que el desarrollo de una aventura debía hacerse casi todo a golpe de ratón. Digo casi todo porque me parecía inevitable la necesidad de tener que codificar parte de la lógica de la aventura por medio de un lenguaje de programación.

Segundo, el lenguaje de programación debía ser fácil de aprender y debía incorporar ciertas características de orientación a objetos. Decidí que un lenguaje similar al BASIC sería lo más adecuado. Además decidí que debía ser un lenguaje de programación que tuviese los elementos suficientes (al menos los que yo considero suficientes) para desarrollar aventuras. Características como bucles, procedimientos y toma de decisiones serían implementadas.

Tercero, en principio el desarrollo del sistema estaría orientado a poner en mis manos una herramienta que me permitiese diseñar aventuras de forma rápida y sencilla.

Opté por hacer un entorno de programación de aventuras visual para el sistema operativo Windows ya que era el que me proporcionaba las herramientas más adecuadas para lograr con éxito los objetivos marcados. Además opté por el modelo compilador-intérprete que es el que ha demostrado mayor éxito en el desarrollo de aventuras.

Una cuestión importante de diseño fue el mantener cierta similitud con el PAWS. En Visual SINTAC se pueden encontrar el vocabulario, similar a la tabla de vocabulario de PAWS, así como las localidades o los objetos. En cambio hay cosas nuevas como soporte para la creación de PSIs.

Esta vez decidí usar como lenguaje de desarrollo el Visual Basic. Un lenguaje infravalorado por muchos programadores pero que a mí me ha demostrado ser lo suficientemente válido para proyectos de toda índole. Evidentemente la velocidad de ejecución que obtendría no sería tan elevada como si hubiese decidido usar C. Pero ¿importa esto realmente en el caso de aventuras conversacionales?. Una ventaja de usar Visual Basic frente a C es que el tiempo de desarrollo de Visual SINTAC disminuiría notablemente y me permitiría obtener una versión funcional en poco tiempo. Una desventaja es que el Visual SINTAC no sería portable a otros sistemas. En fín, consideré que las ventajas superaban a los inconvenientes.

Una vez marcadas las pautas iniciales me puse manos a la obra, rondaban los primeros meses del 98, y en poco menos de dos meses obtuve una versión que ya incorporaba las herramientas visuales básicas (editor de vocabulario, de localidades, de objetos, de PSIs y de mapas).

Debido a diversos motivos el proyecto estuvo parado unos cuantos meses y hacia finales del 98 lo retomé. A principios del 99 ya tenía un rudimentario intérprete de un lenguaje de programación similar al BASIC funcionando, y además a una velocidad aceptable.

Pocos meses después, hacia mediados del 99, ya había una versión del sistema funcional. Aún quedaban por pulir algunos detalles y, sobre todo, corregir la ingente cantidad de errores que seguro surgirían. Debo decir, como anécdota, que el proyecto sufrió un gran empujón debido a que en aquella época me encontraba desplazado fuera de mi ciudad por motivos de trabajo. Pude dedicar gran parte de las tardes durante tres meses (mi trabajo me permitía tener las tardes libres) a codificar el sistema en la habitación del hotel donde me hospedaba.

Visual SINTAC, además, incorpora capacidades multimedia. El apartado gráfico queda cubierto con el soporte a los principales formatos: JPEG, BMP, entre otros. En cuanto al sonido soporta formatos WAV, MOD, S3M, XM e IT. Respecto al soporte de sonido, al principio decidí usar las librerías FSOUND. Debido a ciertos problemas detectados en algunas configuraciones, actualmente Visual SINTAC incorpora las librerías MIDAS de sonido.

A fecha de hoy, finales del 99, está casi lista la primera versión del Visual SINTAC. Creo que he cumplido los objetivos marcados con creces. Sobre todo he conseguido un sistema que me permitirá crear mis futuras aventuras de forma rápida y eficaz.

Además me gustaría pensar que he sentado las bases que marcarán el futuro de los sistemas de creación de aventuras conversacionales: entorno totalmente visual, lenguaje de programación potente pero fácil de aprender, orientación a objetos.

Pero Visual SINTAC no se quedará estancado. Tengo planes a medio plazo que, en cierta medida, mejorarán el sistema. Uno de ellos es reprogramar el intérprete en un lenguaje más portable. He decidido usar C++ por razones obvias como que el código que realice podrá ser recompilado en cualquier sistema que disponga de un compilador de este lenguaje. Además pretendo usar las librerías Glk para implementar los mecanismos de entrada y salida. Esto no comprometerá la portabilidad del intérprete ya que las Glk están disponibles para la mayoría de sistemas.

Además vendrán otras mejoras que aún están por decidir. Quizá implemente mecanismos más potentes como la incorporación de eventos en las localidades, objetos y PSIs.

En fin, en Visual SINTAC pretendo reflejar lo que yo considero debería ser un sistema de creación de aventuras conversacionales moderno. ■

LOS ROGUELIKE

Por: Jlceb


mediados de los 70, dos estudiantes californianos llamados Toy y Wichman jugaban con una nueva librería llamada «curses» que proporcionaba a los programas la revolucionaria posibilidad de posicionar el cursor en pantalla. Al igual que muchos otros programas surgidos alrededor de una situación similar por aquél entonces, el resultado se convirtió en una leyenda.

Aunque eran grandes fans de Adventure, sabían que los juegos de aventura presentaban un mundo predefinido, igual para cada partida. Su objetivo, en cambio, fue hacer un juego que fuera distinto cada vez que jugaras a él.

El resultado se llamó «Rogue». Empleaba los caracteres ASCII del terminal y las nuevas posibilidades del posicionamiento del cursor para «dibujar» una mazmorra aleatoria en la que el jugador desciende nivel tras nivel atravesando pasillos y salas, enfrentándose a una variedad de monstruos y trampas y empleando diferentes armas, hechizos y objetos que podía encontrar en el laberinto, o derrotando algunos monstruos. Rogue incorporaba un sistema de características basadas en el rol, como puntos de vida, hambre, etc... El jugador podía desplazarse mediante las teclas de desplazamiento del cursor, y realizar acciones como comer, leer, abrir puertas o atacar monstruos, mediante teclas predefinidas.

Rogue tuvo una historia bastante larga. Eventualmente se convirtió en un juego comercial, con versiones gráficas para Amiga o el Spectrum. Sin embargo, el hito fue marcado por la versión original y su sencillo concepto de emplear caracteres ASCII para identificar al jugador, monstruos, paredes y objetos del mundo.

Rogue fue, indudablemente, el primer RPG de la historia. Sin duda podría trazarse una línea desde él a cualquier otro RPG de la actualidad. Pero es otra línea distinta la que nos interesa hoy.

MORIA

La popularidad de Rogue, que fue incluído entre una de las primeras versiones de Unix, llevó irremediablemente a la aparición de diversos clones. El primero de ellos fue Moria, escrito en Pascal en 1983 por Robert A. Koeneke. A diferencia del Rogue original, Moria

era gratuito, y estaba levemente influenciado por la obra de Tolkien: el objetivo consiste en introducirse en Minas Moria y acabar derrotando al Balrog. A diferencia del Rogue original, el jugador puede elegir una raza y profesión, incrementando la variedad del juego.

Pero la novedad más importante en Moria fue la disponibilidad del código fuente, que marcó la diferencia con Rogue y sirvió para dar el pistoletazo de salida a todo un género de juegos: los «roguelike» o «juegos como Rogue».

ANGBAND

Moria fue portado a C por varias personas, entre ellas James E. Wilson, cuya versión (UMoria) sirvió de base a una versión muy, muy ampliada de Moria llamada Angband. creada por Alex Cutler y Andy Astrand en 1991. Para muchos, el Angband fue su primer juego roguelike: añade a Moria gran cantidad de monstruos, trampas, así como monstruos únicos, artefactos como el Anillo Único con sus propias características y comportamiento, e incluso una ciudad donde comprar y vender objetos. Angband es un juego monumental y acabarlo sin hacer trampas es una tarea impresionante.

Siguiendo la línea marcada por el Rogue original, prácticamente ningún roguelike proporciona la opción de grabar partida (aunque si abandonamos el juego y volvemos más tarde, la partida continúa donde la dejamos). Se supone que el jugador debe ser lo bastante cuidadoso como para no morir, y acabarse el juego sin romper esta regla es realmente difícil pero definitivamente posible.



De Angband han surgido incontables variantes (¡y recordemos que Angband era ya una variante de una versión de un clon de otro juego!) siendo la más popular, Zelazny Angband (ZAngband), creada por Topi Ylinen en 1994 y añadiendo mucho, mucho más sobre la base de Angband: docenas de clases (muchas con habilidades propias, como vampiro o espectro), misiones, múltiples ciudades, bosques, castillos... ZAngband es Angband llevado a la próxima generación... Aunque volveremos a esto más adelante.

NETHACK

Retrocedamos un poco en el tiempo, aproximadamente hasta poco después de la aparición de Moria. Hack fue una versión mejorada de Rogue, que no se tomaba muy en serio: incluía clases de personaje tales como arqueólogo, hombre de las cuevas o turista (con cámara y camiseta hawaiana incluída), tiendas (donde se puede robar), así como cantidad de nuevos monstruos, sortilegios y objetos. El Hack original fue escrito por Jay Fenlason en 1985, con ayuda de diferentes personas.

Como es natural, surgieron bastantes variantes de Hack. Mike Stephenson recogió varias de ellas y las fusionó y amplió en una nueva variante llamada Nethack. Desde entonces, Nethack ha sido mantenido por toda una serie de personas, colaborando a través

de Internet (de ahí su nombre). Es uno de los roguelike más populares, ya que a la mazmorra aleatoria añade ciudades, misiones y niveles predefinidos, y en definitiva tiene un objetivo y una especie de argumento que se desarrolla a medida que avanza el juego.

ADOM

Adom es el roguelike más joven, nacido en 1994 de la mano de Thomas Biskup, quien todavía mantiene el desarrollo del juego activo. Se puede considerar la siguiente generación en los roguelike. Adom se desarrolla en un mundo definido, Arcandia, con sus costumbres, ciudades y personajes claramente diferenciados. Tiene un argumento, un desarrollo, situaciones a resolver y personajes con los que hablar e interactuar. La complejidad de Adom es también un punto aparte respecto a los juegos clásicos: desde habilidades específicas para cada tipo de arma, a la posibilidad de escoger una táctica específica de combate.

PERO... ¿QUÉ SON LOS ROGUELIKE?

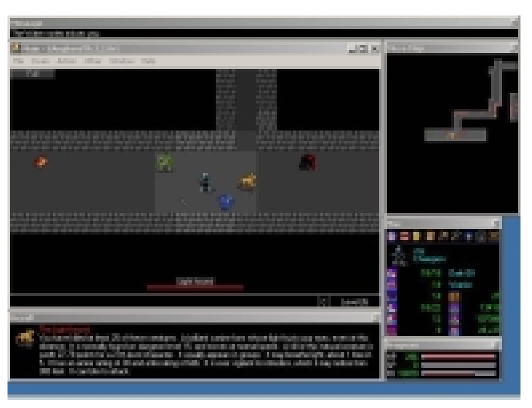
Todos los géneros se definen por su historia. Un juego determinado se califica como una aventura, y no como un simulador, porque es parecido o está basado en otro juego anterior que era una aventura. Podríamos definir muchas de las características que tienen en común los juegos que hemos analizado:

- Juegos de rol con características
- Un sólo jugador, manejando a un único personaje
- Basado en turnos (no en tiempo real)
- Gráficos basados en caracteres ASCII
- Una o varias mazmorras pseudo-aleatorias
- Movimiento y comandos basados en pulsaciones de tecla
- El personaje incrementa sus habilidades y nivel mediante la práctica
- Son necesarios días, a veces meses, para acabar el juego
- Gran variedad de objetos y acciones (abrir, comer, leer, vestir, etc...)
- Basados en su mayor parte en combatir y matar monstruos
- Se emplean muchos años (de 5 a 20) para hacer un roguelike completo
- Juegos portables, con versiones para múltiples plataformas

Rogue y Moria, los primeros roguelike, emplean un sistema de características basado en Dungeons & Dragons, el juego de rol. Muchos otros roguelike se han inspirado en ellos, así que es habitual que nuestro personaje cuente con características como Sabiduría, Carisma o Destreza, que se involucran en las acciones que realiza (por ejemplo, el combate, o lanzar hechizos) y pueden modificarse durante el juego mediante la práctica o la acción de objetos mágicos, hechizos, y ataques de algunos monstruos.

Otros factores que deben controlarse habitualmente, además por supuesto de las heridas y su curación mediante el descanso, hechizos u otros medios, son el hambre (suele abundar comida de distinto tipo, o poder comprarse; varios roguelike permiten comer y cocinar cuerpos de monstruos caídos), la alineación (realizando acciones fraudulentas o malvadas puede variarse en algunos RPG) y factores como lo bien que le caemos a nuestro Dios (a menudo podemos rezar o ofrecerle sacrificios).

¿QUÉ TIENEN DE BUENO?



Tras esta introducción deberías saber exactamente qué es un roguelike, qué os encontraréis delante de una pantalla y cuál ha sido su historia y evolución desde su nacimiento hasta nuestros días.

Pero falta la esencia, el jugo, la explicación de por qué un juego en el que desplazamos una arroba por pantalla matando letras y números de colores puede llegar a pegar a una persona delante de una pantalla durante meses. No tengo una explicación a eso, de la misma manera que no tengo ninguna explicación para las personas que se maravillan de que alguien pueda jugar a un juego que no consta de otra cosa que texto en pantalla que hay que leerse y donde las acciones a realizar deben teclearse con todas las letras y palabras.

Pero la profundidad que llegan a alcanzar unos juegos que tras pasar de mano en mano llegan a tener un período de desarrollo no ya de años sino de décadas, deja atrás a cualquier otra cosa imaginable. ¿Qué juego hay, aparte del Nethack, con un comando que sirve para limpiarse las orejas? No puede cogerse un roguelike como un entretenimiento para pasar el rato durante 10 minutos: casi cada tecla es un comando diferente que conviene conocer. La curva de aprendizaje de un roguelike es terrible: quizá hacen falta semanas para conocer un juego hasta el punto de disfrutarlo realmente. Pero el resultado, definitivamente, vale la pena. ■



DEFINITIVE SOFTWARE

Por: Jaba



efinitive Software, la compañía madrileña que obtuvo el "Premio a la Mejor Compañía Aventurera" otorgado por el Year Zero Club en 1995, ha vuelto a la escena aventurera tras más de un año de completa inactividad. Tiene unos objetivos claros definidos para el final de milenio que pasamos a enumerar en exclusiva primicia...

NUEVA VERSIÓN DEL DAP

En primer lugar, Definitive acaba de presentar la nueva versión del parser DAP (Diseñador de Aventuras Profesional), la 5.10. Esta nueva versión tiene nuevas características, entre las que cabe citar:

- ✓ Un reconocedor de sintaxis potencialmente superior
- ✓ Un sistema de edición de procedimientos muy mejorado: ahora se edita a pantalla completa (usando un editor de texto externo); en las versiones anteriores se usaba un editor interno de líneas.
- ✓ Menores requerimientos de sistema (ahora requiere cuatro veces menos memoria y se ejecuta en más equipos)
- ✓ Un driver (experimental) de modo de alta resolución para las tarjetas Voodoo 3 y Voodoo Banshee (según Definitive, quizás podría funcionar en otras tarjetas)
- ✓ Una documentación mucho más clara.

Es preciso destacar que DAP, que desde su primera versión (lanzada en 1989) ha sido un producto shareware, Definitive decidió convertirlo a freeware tras la celebración de la *Madridaventura'99*.

La página principal del DAP es

http://www.geocities.com/siliconvalley/grid/2294/dap.htm

LAS CLASES DE AVENTURAS

Se trata de un conjunto de clases en C++ pensado para facilitar la creación de aventuras conversacionales en este lenguaje; según Definitive implementará un potente sistema orientado a objetos que podrá soportar incluso partidas multijugador. *No se trata de un parser, sino de una librería de programación.* Estas clases serán de libre distribución, y deberían poder compilarse en distintos entornos.

Sabréis más de estas clases (cuyo nombre aún no se ha decidido) en este mismo número, pues su autor ha decidido ir desarrollándolas en una serie de artículos en este mismo fanzine.

Definitive tiene una versión alpha experimental a disposición de quien desee evaluarla en:

http://www.geocities.com/sytovos/cav.zip

MENZOBERRANZAN

Es el nombre de la aventura que Definitive está preparando; aunque posiblemente varie su nombre debido a la coincidencia con un antiguo juego de rol de Black Isle. Se trata de una aventura ambientada en el universo de los Reinos Olvidos, en la ciudad de elfos oscuros del mismo nombre.

Aún no se conoce demasiado del argumento; actualmente Definitive está buscando colaboradores para el desarrollo del programa.

AVENTURA CONMEMORATIVA MADRIDAVENTURA'99

Se trata de una pequeña aventura que ha desarrollado Definitive Software con motivo de la convención aventurera; se puede encontrar ya disponible en

http://www.geocities.com/sytovos/avent.zip

PAPADA SOFT

Por: ManoWar

Bueno, con este dossier voy a tratar de explicaros las andaduras de este singular grupo de programadores de aventuras, aquí trataremos desde los inicios del grupo hasta el presente y el futuro más cercano. Pero, sin enrrollarme más vamos con ello.

Toda la historia se remonta al año 1990, en efecto, la época del Spectrum y sus 8 bits. Fue aproximadamente en ese año cuando Luis Felipe Morales Bendicho adquirió el programa PAWS (The Profesional Adventure Writing System). PAWS es un programa para la creación de aventuras conversacionales en ordenadores de 8 bits. Teniendo en su poder el programa comenzó a desarrollar pequeñas aventuras y aprovechando que tenía un par de colegas de su barrio aficcionados a las conversacionales pues montó lo que mucho más tarde daría lugar a Papada Soft, luego esto nos lleva a que los fundadores del grupo fueron:

- Luis Felipe Morales Bendicho.
- Jorge Palencia Gallardo.
- Miguel Angel Segura Bermudo.
- Jose Rodriguez Ruiz.

De toda esa época salieron las aventuras: «Cadena Perpetua», «La Conquista De Venus», «El viaje de Colón», «Venganza Mortal» y «El Libro Encantado». Desafortunadamente la única que se ha podido rescatar ha sido «Venganza Mortal». Sea como sea todas eran bastante malas y salvo «Cadena Perpetua» y «La Conquista De Venus» las demás fueron meras pruebas. Esperamos que algun dia podamos disfrutar de ellas.

Justo despues de esto llegó «la muerte del Spectrum» y con ella la disolución del grupo durante 5 largos años. En todos esos años el grupo quedó virtualmente desconectado del mundo de la aventura. No sería hasta el año 1995-1996 cuando nuevamente Luis Felipe Morales adquirió un PC y trasteando con un CD de una revista encontró el parser NMP. Cual sería su sorpresa al encontrarse después de tanto tiempo con un parser, y lo mejor de todo es que era parecidísimo al PAWS, motivo de más para ponerse manos a la obra. Asi pues nuevamente Luis Felipe, Jorge y Miguel formaron Papada Soft y de esta formación salieron los juegos: «Caso Cerrado», «Caso Cerrado II» y «Me Marcho Pa' Francia 98". Ya a último tiempo de finalizar «Caso

Cerrado II» llego la internada en Internet y con ello la revolución al grupo.

Estamos ya sobre el año 1998-1999. Una vez cogida la conexión de Internet, las ganas de Miguel y Jorge comenzaron a menguar, debido generalmente a motivos personales y laborales, mientras que las ganas de Luis Felipe continuaban en aumento y comenzaba a plantearse miras más altas, ahora sus pensamientos estaban plantados en una Aventura Gráfica. La creación de una aventura gráfica siempre ha sido difícil y eso bien lo sabía Luis Felipe por lo que en principio y sin miras muy altas comenzó la creación del nuevo grupo. El primero en añadirse al grupo y que podríamos identificar como cofundador de esta nueva etapa fue: JumboJr o lo que es lo mismo Juan José Perera. Al cual se le dió la oportunidad de cambiar el nombre del grupo o dejarlo tal cual estaba, el resultado salta a la vista ;-).

En dos dias el grupo ya estaba constituido por cuatro personas, las cuales eran:

- Luis Felipe ([Manowar]).
- Juan Jose Perera (JumboJr).
- Roger Olier Alvira (Aigam).
- Pablo Jimenez Martínez (Dekar).

La base de creación del grupo había sido: crear una aventura gráfica en la que el protagonista sería un clérigo. Esa fue la base de esta nueva etapa. De todo eso, y después de tirar dos o tres guiones a la basura, salió el proyecto «Fray Luis» que a grandes rasgos versa sobre un fraile que va a cumplir 18 años y debe ser nombrado oficialmente fraile.

Más tarde comenzó la adhesión de grafistas, el primero en entrar fue Marcos Romero (Wingman), el cual ya había colaborado en el proyecto «La isla» y por unos u otros motivos tuvo que desaparecer. Después Marcos cogió a Javi Agenjo (Tamat), otro grafista. Por cierto ambos buenísimos. Después vino un músico: Alejandro Monzo (RAUKO), un guionista: Rafael Guitiérrez Arias (Fortunato) y un programador de IndyJava: Sebastián Alejandro Florin (Tecnosebas).

Siendo ya un grupo bastante robusto se comenzó con la creación del guión del proyecto «Fray Luis». Paralelamente Luis Felipe estaba colaborando con Eduardo Llerena Fernández (Maas) en su proyecto de Estar Guars, una aventura parodiando a la legendaria película Estar Guars. En ese momento surgió la idea de adherir ese proyecto al grupo y así fue, fue en ese instante cuando Maas entró en el grupo y con él su proyecto.

Mas tarde se unieron Tecnosebas, Laian y Vital y aqui finaliza la formación actual del grupo. Es decir que a estas alturas el grupo cuenta con un total de 12 miembros, pero la formación está en constante cambio, varias personas más están en prueba y muchas otras pendientes de entregas de trabajos, pero en definitiva 12 personas ya son un grupo notablente cualificado para llevar a cabo los tres proyectos que tienen en mente.

Entre sus proyectos nos los podemos encontrar de tres tipos:

- **Proyectos acabados:** Un total de 3, si contamos la Beta de Caso Cerrado II. Estos tres proyectos son Caso Cerrado I y II y Me Marcho Pa' Francia 98. Todos aventuras conversacionales y realizados durante la primera etapa del grupo. Caso Cerrado I y Caso Cerrado II tratan sobre las andanzas de un detective que tiene que



investigar un caso extraño. Me marcho pa' Francia 98 es una crítica (en forma de aventura) al ex-seleccionador nacional Javier Clemente, con ella Luis Felipe quiso mostrar sus pensamientos en cuanto al fracaso de la selección española en el mundial.

☐ **Proyectos en curso:** Otros 3, que son el proyecto «Fray Luis», el Estar Guars y la última parte de Caso Cerrado, «Caso Cerrado III: en cuerpo y alma». Los dos primeros son aventuras gráficas. En una tomarás el papel de un fraile que va a ser nombrado fraile «oficialmente» y en la otra tomarás el papel de los personajes más significativos de la película Star Wars, todo ello aderezado con los «gags» y situaciones cómicas que toda parodia puede ofrecer. El último de sus proyectos en curso es una aventura conversacional con la que se cerrará la trilogía Caso Cerrado, en ella tendremos oportunidad de ver como ese detective que ha pasado con nosotros muchas situaciones de tensión consigue cerrar definitivamente el caso de «la mansión» y también podremos observar como algunos de los personajes secundarios vuelven a la carga, esta aventura también nos guarda alguna que otra sorpresa.

☐ **Proyectos futuros:** Este es el ámbito mas oscuro debido a que estos proyectos futuros cambian cada dos por tres. Aun no tienen claro que es lo que les depara el futuro y barajan varias posibilidades, desde otra aventura gráfica al estilo 2D pasando por un juego de rol o incluso una aventura en 3D tipo Alone In The Dark. Como proyectos futuros también podríamos incluir aquí las futuras reediciones de Caso Cerrado I y II a las que se les pretenderá dar una mayor definición de localidades y guión tratando de dejar los cabos sueltos atados de una vez por todas, y también se mudan a las plataformas modernas.

Otro aspecto a destacar, pero del que de momento no se tienen muchos datos, es que se ha comenzado la formación de un subgrupo dentro de Papada en el que todos los miembros son de Madrid. La idea es que este subgrupo se encargue del proyecto Estar Guars con la consiguiente rapidez al no depender de Internet. De momento el subgrupo está formado por 5 personas, pero como ya os digo aún no hay datos fiables.

Pues eso es todo respecto a este grupo sin más que esperar a que sus proyectos vean la luz y poder disfrutar de ellos :-D.

Nota Curiosa: ¿Alguno os habéis preguntado porque se llaman Papada Soft?, he aquí la respuesta. La primera formacion del grupo tenía un amiguete bastante «grande», su nombre es Mariano Rodríguez. El dia que el grupo estaba decidiendo el nombre, a uno de sus integrantes le vino a la cabeza la papada de su amigo, si a esto le añadimos algo relacionado con la informática, como puede ser el Software, pues... ¡alehop! ya tenemos nombre de grupo, Papada Soft. ¿Curioso verdad?.

Nos vemos en el próximo número. ■

El

CAAD

NECESITA tu colaboración.



No lo olvides: los socios hacen el CAAD, los socios **son** el CAAD. Siquieres hacer fanzine, si quieres formar parte de él, no lo olvides: **colabora.**

El CAAD eres tú.

EL ARTE DE LA FICCIÓN INTERACTIVA

Por: Roger Amich



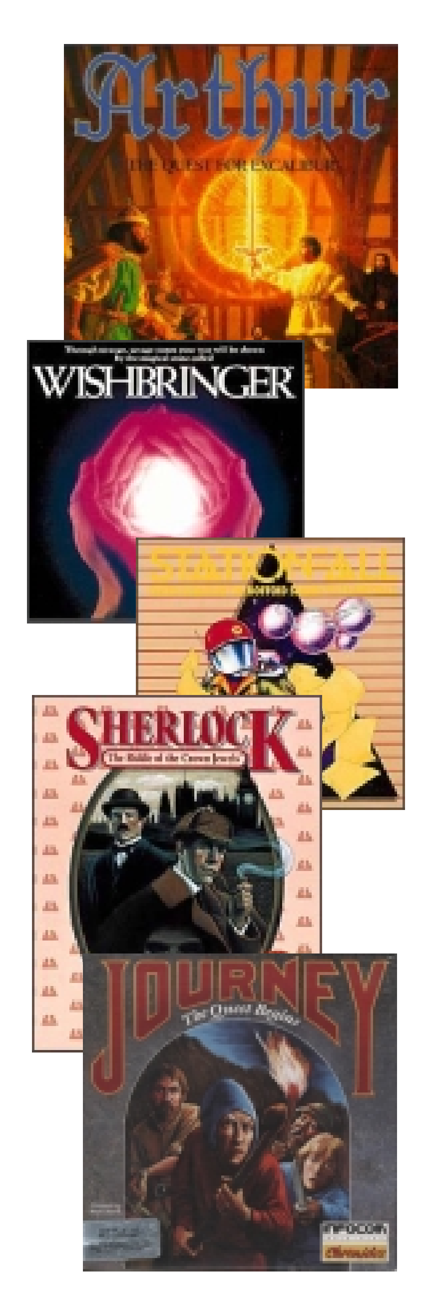
Es de sobras conocido que la historia de la ficción interactiva (o de las aventuras conversacionales, como se prefiera) empezó en los primeros 70, cuando los americanos Willie Crowther y Don Woods crearon "Adventure", un juego en FORTRAN consistente en la exploración de una enorme caverna llena de puzzles y objetos mágicos.



Adventure" es sin duda el juego que más ha marcado la historia de la ficción interactiva. En sus primeros años todas las creaciones fueron versiones más o menos libres del original: "Adventureland" (Scott Adams, 1978), "Zork" (Infocom, 1980), "Colossal Adventure" (Level 9, 1983) o "La Aventura Original" (AD, 1988). Todas incluyen un sistema de cavernas, un oso, una lámpara de batería limitada, un depósito de tesoros... Pero la influencia de "Adventure" fue mucho más profunda. La ambientación tolkeniana y la orientación hacia la resolución de puzzles han sido, hasta ahora, mayoritarias en el mundo de la FI.

La aventura española está aún anclada en esta tradición, sobre todo por el hecho que AD, con sólo seis juegos en su haber y su tardía aparición, no pudo ir más allá. Todos sus juegos están tallados con el mismo patrón: un aventurero impersonal como protagonista, unos objetos a manipular para llegar a un objetivo y unos puzzles del tipo "dar algo a alguien a cambio de algo" o "usar algo con algo".

En el mundo anglosajón, en cambio, las cosas



han ido de manera muy diferente, y la causa la tiene la compañía más importante que existió en los 80: Infocom.

Infocom creó más de 20 juegos, y aunque empezó con el clásico "Zork" y sus secuelas, pronto se diversificó hasta abarcar todos los géneros: el cuento de hadas en "Wishbringer" (Brian Moriarty, 1985), el misterio en "Deadline" (Marc Blank, 1982), el terror en "The Lurking Horror" (Dave Lebling, 1987), el humor en "Leather Goddesses of Phobos" (Steve Meretzky, 1986), los mundos alternativos en Trinity (Brian Moriarty, 1986)...

Todos estos juegos compartían una detallada ambientación, descripciones literarias, protagonistas y personajes no jugadores complejísimos... y aunque contaban con puzzles (pues no dejan de ser uno de los componentes principales de la IF), éstos se abordaron desde otro punto de vista: su integración en la historia y el argumento.

Actualmente la *Interactive Fiction* anglosajona, totalmente en manos de los aficionados, ha avanzado mucho más, hasta ser considerada una forma de arte y llegar a unos niveles de virtuosismo nunca vistos. No en España, donde aún estamos anclados en el pasado.

El arte de la ficción (interactiva) es el título que toma esta sección que ahora empieza. En ella voy a intentar explicar lo que pasa en el mundo anglosajón, no a nivel de programación o *parsers*, sino de argumentos, estilo, filosofía, tendencias... En cada capítulo voy a analizar un aspecto concreto de la FI, poniendo ejemplos de diferentes juegos.

Mi intención es doble: mostrar lo que está pasando allí y ayudar a importarlo hacia aquí. Creo sinceramente que si la aventura española quiere sobrevivir debe evolucionar, y aunque debemos tomar nuestro propio camino, la *Interactive Fiction* es buen modelo a seguir. ■



GLK
¡Y NO REINVENTES LA RUEDA!

Por: Zak McKraken

Glk es una librería de entrada/salida, que proporciona funciones para las operaciones más habituales de las aventuras conversacionales (como mostrar texto rompiendo automáticamente las líneas, gráficos, dividir la pantalla en regiones o ventanas, etc.)

En este artículo se da una visión general de las etapas de programación que se requieren para la creación de un parser o un juego conversacional implementado directamente en C, indicando en qué punto del proceso sería útil Glk.

Se explica también la filosofía que se persiguió en el diseño de Glk y las ventajas que conlleva usarlas, para terminar finalmente con una reseña del estado actual de desarrollo de Glk y lo que se ha hecho con ella.

A quién puede interesar este artículo:

- A desarrolladores de «parsers».
- A programadores de aventuras conversacionales que no quieren usar parsers y prefieren programar en C o C++.
- A los interesados en la «multiplataforma», es decir, que sus juegos puedan jugarse sin modificaciones tanto en un PC con Windows, como en un PC con Linux, pasando por un Macintosh y otros ordenadores.
- A todo interesado en conocer los entresijos de como se hace un parser.

A quién no interesará:

- A jugadores de aventuras conversacionales, que no planean programar una. No necesitan saber nada de Glk para jugar.
- A programadores de aventuras conversacionales que usen NMP, SINTAC, Caecho o SKC y no piensan aprender a usar otro parser, ya que estos parsers no usan ni usarán Glk.

INTRODUCCIÓN

Reinventando la rueda, el chasis, el volante y el salpicadero

Supongamos que estás pensando crear un nuevo parser para las conversacionales. Te enfrentarás a los siguientes retos:

❶ Inventar un lenguaje de programación (la invención no necesita ser completa, puedes copiar ideas de otros lenguajes existentes). Si lo prefieres puedes no inventar nada en este paso y usar un lenguaje existente como C o Java.

❷ Escribir, usando el lenguaje que has inventado en el paso 1), toda una librería de apoyo al desarrollo de aventuras. Esta librería incluirá funciones útiles al programador de aventuras y definiciones de tipos de datos y tipos de objetos habituales en las aventuras. En mi opinión este es el paso donde se debería hacer un mayor esfuerzo, y sería el factor diferenciador entre tu parser y los otros previamente existentes.

❸ Inventar una máquina virtual, es decir, un conjunto de códigos básicos que tu sistema será capaz de ejecutar (algo así como el conjunto de condactos del PAWS, o el conjunto de instrucciones máquina de la máquina Z, o el conjunto de bytecodes del Perl). No necesitas inventar una máquina virtual nueva, puedes elegir una de las muchas que ya existen, Java incluida.

O si lo prefieres puedes puedes pasar completamente de máquina virtual, en cuyo caso tu parser sólo servirá para una máquina real concreta, la que hayas elegido como objetivo (un PC con MS-DOS, por ejemplo, o un PC con Windows98, o tal vez ambas si estás dispuesto a duplicar esfuerzos).

❹ Hacer un compilador que convierta el lenguaje de programación que has inventado en el paso 1 en el código máquina que utilice la máquina virtual del paso 3. Si en el paso 1 elegiste un lenguaje ya existente (por ejemplo C) y en el paso 3 elegiste una máquina ya existente (por ejemplo un Pentium con Linux), el compilador ya existirá también (por ejemplo gcc), por lo que puedes saltarte este paso.

Si el lenguaje que has inventado en el paso 1 es de tipo visual, el compilador necesitará también un interface gráfico que permita al programador de juegos elegir componentes básicos y relacionarlos entre sí. Evidentemente este interface sería bastante dependiente de la librería para aventuras creada en el paso 2.

❺ Hacer un intérprete que tome el código máquina que se ha generado en 4 y lo ejecute. Si habías elegido en 3 una máquina real, no necesitarás intérprete, ya que el código podrá ser ejecutado directamente por esa máquina real (o un emulador de la misma).

Este intérprete será quien finalmente lea lo que escribe el usuario y se lo pase al parser, el cual elaborará una respuesta (texto, o gráfico, o multimedia) y se la enviará al intérprete para que la muestre en pantalla. De modo que este punto 5 aún se subdivide en las siguientes tareas:

5.1 Desarrollar funciones que puedan leer comandos escritos por el jugador, o tal vez movimientos del ratón, o de un lápiz como en los ordenadores de mano, para pasárselos a la máquina virtual.

5.2 Desarrollar funciones que puedan mostrar texto en la pantalla. Piensa que no todos los sistemas tienen printf, ni siquiera aunque sean programables en C (un buen ejemplo de esto es la Palm Pilot).

5.3 Desarrollar funciones capaces de abrir ventanas, cambiarlas de tamaño, mostrar texto en ellas, hacer scroll, mostrar gráficos, etc...

Los tres puntos anteriores debes repetirlos para cada posible plataforma en la que quieres que se ejecuten tus juegos. Windows, Macintosh, Linux, DOS, Palm Pilot, ... Cada una tiene sus propias ideosincrasias. Sería muy interesante que todo lo relacionado con la entrada/salida estuviera claramente separado del resto del intérprete, para no tener que reprogramar el intérprete completo para cada nueva plataforma, sino sólo la parte relacionada con E/S.

Como ves, es una ingente labor. A menudo los programadores son aficionados a reinventar la rueda, y acometen los cinco pasos anteriores una y otra vez. Algunas excepciones notables a esta tendencia son:

- **Juegos desarrollados sin parser**. Se programan directamente en un lenguaje (pongamos el C), por lo que se saltan los pasos 1, 3 y 4 (la creación del lenguaje, la máquina virtual y el compilador). Por desgracia suelen saltarse también el paso 2 (la librería para aventuras) por lo cual el código desarrollado no suele ser utilizable para otro juego distinto. Por otra parte, al no usar máquina virtual, no requiere la programación de un intérprete, sino que el propio juego ejecuta directamente las operaciones necesarias y lleva a cabo toda la entrada/salida que necesite. Desgraciadamente la entrada/salida suele estar demasiado mezclada con el resto del juego, por lo que tratar de portar el juego a otro operativo suele ser muy complejo. Si la entrada/salida estuviera claramente separada del resto del juego, adaptar el juego a otras plataformas sería mucho más sencillo, puesto que sólo habría que reemplazar la parte de entrada/salida.
- **Inform**: Se crea un nuevo lenguaje (paso 1), toda una librería escrita en ese lenguaje (paso 2) y un compilador (paso 4) para convertir todo esto en código, pero se usa como máquina virtual la máquina Z. En este caso no se reinventa el paso 3, puesto que la máquina Z ya había sido inventada tiempo atrás por Infocom. Esto tiene la ventaja de que tampoco es necesario trabajo en el paso 5, ya que los intérpretes ya existentes para la máquina Z servirán para los nuevos juegos. (No obstante, otros programadores tomaron el reto de hacer intérpretes nuevos, para ordenadores y sistemas operativos modernos, como Windows95 o Linux).
- **Glulx**: Se crea una nueva máquina virtual, llamada Glulx (paso 3) pero no un nuevo lenguaje de programación. El lenguaje sigue siendo Inform. Ahora es

necesario crear un nuevo compilador de Inform a Glulx (paso 4) y nuevos intérpretes de esta máquina virtual (paso 5), pero no es necesario modificar la librería del paso 2, puesto que estaba escrita en Inform y el lenguaje no ha cambiado (en realidad la librería sí que requiere unas modificaciones pero son mínimas, no es una reescritura desde cero).

- **PAWSEZ**: Este proyecto que nunca fue terminado, tomaba código fuente en PAWS y lo compilaba en código Z. Por tanto se centraba en el paso 4, el compilador, sin reinventar nada del resto. Como lenguaje de programación usaba el ya existente PAWS, y como máquina virtual la ya existente máquina Z. La librería de la aventura (paso 2) era inexistente en PAWS pues formaba parte del código fuente de cada juego. El objetivo era reconvertir los antiguos juegos de Spectrum, pero nada impediría desarrollar nuevos juegos con este sistema. El lenguaje de programación sería PAWS, pero los nuevos juegos podrían ejecutarse en cualquier ordenador (no sólo en el Spectrum) gracias a todos los intérpretes existentes para la máquina Z.

  Observa que teóricamente sería posible también la operación contraria, es decir, utilizar como lenguaje de programación Inform y como máquina virtual el PAWS, y crear un compilador de Inform que generara directamente ficheros SNA ejecutables en un emulador de Spectrum, con el run-time del PAWS original. La utilidad de este enfoque, no obstante, se me escapa.

- **CAV**: es una librería para aventuras (paso 2) escrita para C++, por lo que ni el lenguaje ni el compilador son reinventados. Por otra parte tampoco utiliza máquina virtual, por lo que no necesita intérprete, aunque sí será necesaria la programación de toda la entrada/salida, que será específica de la plataforma de desarrollo.

- **Glk**: Por fin aparece el tema que da título al artículo. Glk es una librería que se ocuparía de la parte de entrada/salida del juego, es decir, las tareas 5.1, 5.2 y 5.3.

GLK: OBJETIVOS

Glk tiene dos objetivos claros:

1. Potenciar la multiplataforma. Si programas un juego directamente en C, habrá dos partes claramente diferenciadas en el juego: la parte que maneja las variables, los objetos, los personajes, etc. y la parte que saca los mensajes por pantalla, lee comandos del jugador, muestra gráficos, etc. Estas partes podrían aparecer mezcladas en el código fuente, pero tenerlas separadas tiene una ventaja evidente: la parte que maneja las interioridades del juego puede ser escrita en C estándar, que compilaría fácilmente en cualquier plataforma, y sólo la parte de la entrada/salida debería ser reescrita para obtener diferentes versiones del juego (versión solo texto para MS-DOS, versión con ventanas para Windows, versión sólo texto para Linux/Unix, versión con ventanas para X Window).

   Glk sería precisamente la parte que hace la entrada/salida, y actualmente ya existe para las plataformas antes nombradas, por lo que si vas a hacer un

juego en C, ya tienes gran parte del trabajo resuelto incluso para diferentes sistemas operativos.

Si tu objetivo no es programar un juego en C, sino un intérprete para que ejecute el código de alguna máquina virtual (ya existente o inventada por tí, es lo mismo), igualmente podrás hacer uso de Glk para la parte del intérprete que realice la entrada/salida. Así puedes escribir un intérprete fácilmente portable a otras plataformas, ya que la parte del intérprete que no hace la entrada/salida podrá escribirse en C estándar.

❷ Permitir la creación de máquinas virtuales «extensibles». Este segundo objetivo es mucho menos obvio que el anterior, pero tal vez de mayor importancia aún.

Imagina que has diseñado un nuevo lenguaje de programación o «parser» (llamémosle por ejemplo Kaos) y una máquina virtual asociada (llamémosla la «maquina K»). Tu máquina K tiene operaciones para pintar un gráfico, para mostrar texto, para cambiar el tipo de letra, para tocar música MIDI, etc. En fin, que le has puesto todas las opciones multimedia que se te han ocurrido. El lenguaje Kaos tiene sus instrucciones de alto nivel para hacer cada una de estas cosas, por ejemplo TocaMusica(), etc. Has escrito también un intérprete de la máquina K, que ejecuta los juegos, y la has enlazado con la librería Allegro, por ejemplo, para poder soportar todas esas maravillas multimedia.

Y ahora alguien va y te dice: «Pero no soporta texto scrollable sobre un fondo gráfico JPG». Y tú dices, «Caramba, es verdad, pero eso se lo añado yo rápidamente». Tendrías que extender la máquina K con una nueva instrucción, o conjunto de ellas; tendrías que modificar el lenguaje Kaos para añadirle un nuevo comando apropiado; tendrías que modificar el compilador de Kaos para que convierta adecuadamente ese nuevo comando a la nueva máquina K; y finalmente tendrías que reescribir una parte del intérprete, para que sea capaz de llevar a cabo las nuevas capacidades de la máquina K.

Y cuando ya lo tienes, alguien va y te dice: «Pero no soporta gráficos in-line al estilo del tag <IMG> del HTML», y se repetiría la historia de modificaciones a tu sistema. Y dentro de unos años, alguien dirá «Pero no soporta frigulillos sónicos 4D en dolby-surround-THX»... Y tal vez tú ya has abandonado completamente el mundillo de la aventura, y no has dejado los fuentes de tu compilador, y nadie será capaz de añadirle esta nueva capacidad a tu sistema.

La filosofía de Glk propone aislar todas esas posibilidades de entrada/salida en la librería Glk, sacándolas de la máquina virtual y del lenguaje de programación. La única instrucción de entrada/salida que debería tener la máquina K sería una «llamada a glk». Esta llamada incorporaría un número que sería el servicio solicitado más todos los parámetros que requiera ese servicio. Así por ejemplo, para mostrar texto, la máquina K podría ejecutar glk(3, txt); suponiendo que 3 sea el servicio encargado de mostrar texto, y txt el texto correspondiente. Para mostrar un gráfico podría hacer glk(8, gr); suponiendo que 8 es el servicio encargado de mostrar gráficos, etc. Cuando el intérprete se encuentre la orden glk(), simplemente llamará a la librería para que lo lleve a cabo.

Si en el futuro deseamos un nuevo servicio (por ejemplo «Activar frigulillos



sónicos 4D en dolby-surround-THX), bastaría añadir este nuevo servicio a la librería Glk, dándole un número de servicio que no estuviera ya usado, por ejemplo el 125 ¡y no sería necesario extender la máquina K, ni el cambiar el compilador de Kaos, ni modificar el lenguaje de programación! ¡Ni siquiera habría que tocar el intérprete, sino que bastaría enlazarlo de nuevo con la nueva versión de la librería! Si tu sistema soporta enlace dinámico, como Windows por ejemplo, bastaría sustituir Glk.DLL en tu sistema. No necesitarías cambiar ni un solo ejecutable.

Con este único cambio, ya podrías hacer juegos con soporte para frigulillos sónicos. Bastaría que tu programa hiciera una llamada a glk(125) cuando requiriera de este servicio.

Del mismo modo se le podría añadir soporte de red, por ejemplo, para crear juegos multijugador on-line. O cualquier otra cosa que nos depare el futuro. Lo importante es que todo ello no requiere modificar el compilador ni el intérprete. Sólo cambiaría la librería Glk.

NOTA: Glk proporciona asimismo servicios para informar de qué servicios soporta. Un programa «educado» debería comenzar por preguntar a Glk si tiene soporte para el servicio 125, y si no lo tiene, resignarse a ser ejecutado sin frigulillos sónicos, o bien terminar la ejecución con un mensaje como «Lo siento, tu sistema está demasiado obsoleto como para poder ejecutar mi super-mega-ultra aventura»)

GLK: PRESENTE

Actualmente Glk es una especificación y una serie de implementaciones. La especificación indica lo que la librería debería hacer. No obstante las implementaciones pueden no ofrecer algunos de los servicios (por ejemplo, la implementación para Unix en terminal texto, no tendrá servicios gráficos).

El nivel de funcionalidad ofrecido por Glk está pensado para ser lo bastante genérico como para no forzar a un interface de usuario concreto, sino que cada operativo pueda implementar Glk con el interfaz más «típico» para ese sistema. Así, Glk para Windows tendría ventanas, tipos de letra, etc. mientras que Glk para DOS tendría otro aspecto totalmente diferente. Cabe pensar incluso en diferentes versiones de Glk para Windows, cada una con un interfaz de usuario diferente. Es como si al juego se le pudieran poner diferentes «pieles» que le hacen aparecer diferente ante el jugador.

En concreto, Glk ofrece funcionalidad para:

- Poder dividir la pantalla en «zonas» o ventanas, para mostrar distinta información en cada una de ellas.
- Mostrar texto en pantalla (recibir un chorro de palabras y formatearlas en líneas dependiendo del ancho de pantalla, el tipo de letra, etc, permitiendo además scroll hacia atrás para ver lo que ha salido antes, pausas automáticas, etc.)
- Leer texto escrito por el usuario, con posibilidad de recuperar comandos previos, editar la línea para corregirla, etc.

- Opcionalmente, poder mostrar gráficos, preferiblemente almacenados en algún formato estándar como JPG o PNG (GIF plantea problemas legales, como todo el mundo sabe, pues es propiedad de Compuserve). Estos gráficos podrían mostrarse en una ventana gráfica específica para ellos, o bien mezclados con el texto, que «fluye» a su lado, al estilo de las páginas Web. No soporta todavía fondos gráficos por detrás del texto.
- Opcionalmente, poder hacer ruiditos, que estarían almacenados en algún formato estándar como AIFF.
- Opcionalmente, poder tocar música, que estaría almacenada en algún formato estándar e independiente de la tarjeta de sonido, como MOD (el MIDI, por desgracia, puede sonar de maravilla en una tarjeta y en cambio hacer sangrar los oídos en otra, el MOD suena más o menos igual en todas).

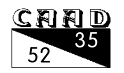
Glk está diseñada teniendo al jugador como protagonista. Todo el aspecto del interfaz de usuario debería ser modificable por el jugador, el cual debe poder elegir su tipo de letra favorito (en los sistemas que tengan varios tipos de letra), la combinación de colores más agradable, el tamaño de la ventana más adecuado a su resolución, etc. Todo esto a veces va en contra de los deseos del programador, quien, a menudo, impone sus gustos sobre el tipo de letra o el color del texto que quiere darle a su juego. Glk permite al programador hacer «sugerencias» sobre estos aspectos, pero el jugador finalmente tendrá la última palabra. Es algo así como lo que ocurre en el diseño de páginas Web: cada navegante puede ver una cosa diferente. El reto del programador en este caso está en que su juego se adapte bien a sus diferentes «encarnaciones» con diferentes «pieles». No hay que perder de vista que en las aventuras conversacionales el elemento clave alrededor del cual debería girar todo es el texto, siendo el resto detalles anecdóticos de los que el juego debería poder prescindir si llega el caso.

Glk está en un estado muy avanzado. Ya se ha logrado todo esto:

- Glk ya ha sido implementada con éxito sobre DOS, Linux/unix (terminal texto), X Window, Windows95/98/2000/NT y Macintosh, aunque el número de servicios soportado en cada caso puede ser diferente. Las versiones más completas son las de Macintosh y Windows, que implementan la especificación completa. Detrás va la de X Window, a la que sólo le falta el sonido. La implementación de DOS no soporta gráficos, pero sí sonido. Finalmente, la implementación para Linux/Unix sobre terminal de texto es la más pobre, pues sólo soporta texto. Eso sí, al menos puede dividir la pantalla en zonas o ventanas.

  NOTA: Existe una versión más de Glk que ni siquiera soporta ventanas, ni divide automáticamente el texto en líneas, ni nada. Simplemente produce un chorro de texto. Está pensada para poder empotrar Glk en un intérprete remoto y de este modo poder jugar vía red.

- Se ha diseñado una máquina virtual que utiliza la llamada a los servicios glk para hacer su entrada/salida, lo cual la convierte como se ha explicado antes en una máquina virtual extensible sin necesidad de redefinirla. Esta máquina es



Glulx. Se ha modificado el compilador Inform para que genere código Glulx, por lo que es posible usar este potente lenguaje para esta nueva máquina virtual. De hecho, se han desarrollado algunos juegos (simplemente demos) usando este lenguaje. Uno de ellos es «Vampiro».

- La librería Glk se ha utilizado para varios proyectos:
  - Desarrollo de un intérprete de código Z. (llamado nitfol)
  - Desarrollo de un intérprete de código Glulx. (llamado glulxe)
  - Desarrollo de un intérprete para la máquina virtual de Hugo (Hugo es otro «parser» anglosajón)
  - Desarrollo de un juego («Dungeon») escrito directamente en C. Este juego utiliza Glk para hacer toda su entrada/salida.

Evidentemente, todos los proyectos anteriores pueden ser compilados en cualquiera de las plataformas para las que existe Glk.

GLK. FUTURO

El futuro de Glk se desarrollará en dos frentes:

- Implementación de la librería para otras plataformas (Windows CE, Palm Pilot, OS/2, etc.)
- Ampliación de las capacidades de la librería. Como ya se ha dicho, las nuevas capacidades podrán ser utilizadas por los juegos inmediatamente, una vez se disponga de la nueva versión de la librería Glk, puesto que no implican cambios en el compilador ni en el intérprete.

El inventor de Glk (Andrew Plotkin) se reserva el derecho de añadir a la librería las funcionalidades que él considere oportunas. Aunque está abierto a sugerencias, retiene su derecho a modificar la librería. De todas formas, puesto que toda la especificación y código fuente de las diferentes implementaciones es totalmente libre, cualquiera puede «copiar» la idea y hacer una librería casi idéntica, reutilizando todo el código y añadiendo nuevas funciones. Andrew Plotkin permite esto, y lo único que pide es que no se llame también Glk a la nueva librería para evitar confusión. De modo que si en España estamos interesados en añadir más cosas a Glk, podemos desarrollar nuestra propia versión y llamarla GlkE, por ejemplo.

En resumen, considero que Glk es una buena idea y que los programadores de parsers y de juegos deberíamos tomar ejemplo de este tipo de iniciativas, tratando de poner algo en común y de hacer código reutilizable. Si la propia Glk no nos gusta, podemos hacer otra más a nuestro gusto, pero partiendo de las mismas ideas. Y si nos gusta, usémosla. Si el esfuerzo que todos dedicamos a esta afición estuviera coordinado de alguna forma, seguramente llegaríamos mucho más lejos. ■

PROGRAMA-CIÓN

UN SISTEMA DE CLASES PARA AVENTURAS CONVERSACIONALES

Por: Javier Basilio Pérez

1: PLANTEAMIENTO DEL PROBLEMA. LA CLASE LOCALIDAD.

Con este artículo da comienzo una serie que pretende construir una *jerarquía de clases* en lenguaje C++ para facilitar la construcción de aventuras. Estas clases se irán diseñando junto a estos artículos, así que es posible (y recomendable) que el posible lector interesado participe y colabore en ellas.

Para poder seguir esta serie, es preciso tener un conocimiento elevado del lenguaje de programación C++, pues se hará uso de características como clases abstractas, funciones puras y virtuales, sobrecarga de operadores y polimorfismo.

Si conoces algo de programación bajo Windows, la idea de las clases de Aventura es similar por ejemplo a la función que cumplen las Microsoft Foundation Classes (MFC) o las Object Windows Library (OWL) para Windows: ocultar el detalle de la programación del interfaz, dejando que las clases se ocupen de engorrosos detalles como los bucles de mensajes de Windows. Nosotros no pretendemos llegar tan lejos, pero valga el símil.

El objeto de todo esto es dar a los desarrolladores que prefieren programar sus aventuras directamente, sin parser, una forma rápida de construir el programa sin tener que preocuparse por muchos detalles relativos al mismo que son repetitivos en la mayoría de las aventuras. Pretendemos además dotar a estas clases de una potencia alta, de modo que puedan servir para hacer cualquier cosa que deseemos; además tenemos dos objetivos prioritarios:

- Portabilidad: las clases deben ser portables y compilables en cualquier entorno. El que escribe irá probando las fuentes en los compiladores siguientes: BORLAND C++ 4.5 (plataforma principal de desarrollo), VISUAL C++ 6 y GNU C++ 2.71 bajo SOLARIS 2.6. Si alguien dispone de otro entorno, agradecería que si surge algún problema me lo comunicara.
- Expansibilidad: las clases deben ser fácilmente expansibles, de modo que sea casi inmediato heredar una de las clases y ampliarla con nuevas funciones o atributos.

Me gustaría mencionar que la idea de estas clases y de una conversacional plenamente orientada a objetos la tomé de una idea que tenía José Luis Cebrián hace algunos años, en la que José Luis Cebrián hablaba de su idea de crear un parser en el que incluso los PSI fueran objetos de igual que el jugador y pudieran hacer lo mismo que él. Estos estarán entre nuestros

objetivos.

Finalmente, indicar que todas las fuentes que publiquemos estarán disponibles en la página del CAAD mientras dure esta serie de artículos. Así mismo, si deseáis colaborar o tenéis alguna duda, podéis contactar conmigo en la lista de correo del fanzine o directamente a mi dirección de e-mail, donde os atenderé gustosamente:

erjaba@teleline.es

Y sin más preludio, comenzaremos el duro trabajo que nos espera.

LA LISTA DE NODOS DOBLEMENTE ENLAZADA

Se trata de un paso previo a todo el desarrollo ulterior, pues nos apoyaremos en ella en el resto de la serie.

Consiste en una clase que implementará una lista de datos de cualquier tipo (para ello, haremos uso de la característica de clases parametrizables de la nueva especificación de ANSI C++). Esto nos será útil para guardar objetos, localidades, etc... Por ejemplo, cuando queramos almacenar los objetos, sólo tendremos que instanciar la clase de la lista para la clase objeto.

Debido al nivel de programación que exige el seguimiento de este artículo, daré por supuesto que en este punto nadie desconoce lo que es una lista de nodos. Si así fuera, estaré gustoso de resolver cualquier problema (de nuevo os remito a mi correo electrónico que ya he citado).

Como queremos hacer las cosas bien, vamos a crear una clase abstracta lista (sin implementación). Después crearemos una clase que llamaremos *dinalista,* que será una implementación con memoria dinámica de las funciones puras (sin implementar) de la clase lista. Así, si en algún momento deseamos expandir las listas o variar su implementación, el procedimiento será muy sencillo.

A continuación enunciamos la clase lista. No implementa todas las funciones de una lista de nodos, pero sí las más comunes y las que a nosotros nos serán necesarias.

CÓDIGO FUENTE DEL FICHERO LISTA.HPP

```
#define TAM 100 // éste es el tamaño por defecto
#define LISTAH BONITALISTA // una cte. para ver si el fichero ya fue incluido

template <class T> class lista  // se trata de una clase parametrizable
{
     public:
          // fuerza la destruccion de los elementos almacenados en la lista
          virtual void matar(void)=0;

          // vaciar la lista
          virtual void flush(void)=0;

          // Insertar un elemento en una posición de la lista
          virtual void aniadir(const T& elem,int n)=0;

          // Suprimir un elemento de la lista
```

```
        virtual void quitar(int n)=0;

        // Acceder a un elemento de la lista (devuelve una copia del mismo)
        virtual void leer(T& que,int n)=0;

        // Devuelve un puntero a un elemento de la lista
        virtual T* leer_referencia(int n)=0;

        // Sobreescribir un elemento existente de la lista

        /* El elemento se pasa por referencia para ganar velocidad. Que no lo
        modificaremos lo aseguramos con el modificador const. Es una ténica
        muy empleada por los programadores de C++ */

        virtual void asignar(const T& elem,int n)=0;

        // Intercambiar dos elementos de la lista
        virtual void intercambiar(int a,int b)=0;

        // Este operador sobrecargado equivale a "leer"
        virtual T operator[](int n)=0;

        // devuelve el número de elementos en la lista
        virtual int cuantos(void)=0;
};
```

Como se puede ver, todas las funciones de esta clase son puras, así que estamos ante una clase diferida (es decir, sin nada de código).

A continuación, veamos algunos ejemplos de utilización. Por ejemplo, supongamos que tenemos una lista de enteros ya creada en el objeto ***m*** y que ***i*** es una variable de tipo entero:

```
m.cuantos() // devuelve el número de elementos en la lista
printf("%d",i[5]); // imprime el quinto elemento de la lista
m.añadir(2,1); // añade el elemento 2 en la primera posición de la lista.
m.quitar(3); // quita el tercer elemento de la lista
scanf("%d",m.leer_referencia(3)); // lee del teclado un número para la posición 3 de lista (observar
que el método leer_referencia difiere del operador sobrecargado [] en que devuelve un puntero).
```

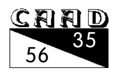
Ahora resta implementar la lista. Esto es cuestión de cada uno, y yo aquí incluiré una implementación bastante hábil que fue diseñada completamente por el que escribe (modestia aparte). Se trata de una implementación dinámica, pero con una ingeniosa técnica que puede acelerar el acceso a los elementos de la lista en varios órdenes de magnitud. ¿Habías oído hablar alguna vez de una lista con caché de accesos?

Como no es mi propósito explicar ahora cómo se implementa una lista, deberás ser tú el que ojee el código si te interesa. Desgraciadamente creo que no será muy claro de seguir, y con certeza está plagado de "chapuzas" y "parches", pero parece funcionar sorprendentemente bien. Si tienes alguna cuestión relacionada con esta implementación, escríbeme.

Posiblemente te traiga sin cuidado el asunto de la implementación de la lista, en ese caso sólo tienes que incorporar el código a tu aplicación final sin preocuparte de nada.

Observaciones:

- ¡No descarto la existencia de posibles bugs en esta versión, aunque ha sido muy depurada!



- La lista exige que el objeto para el que se instancie tenga un constructor de copia y un operador de asignación que realice una copia en *profundidad.* Lo que quiere decir que ambos deben encargarse de crear réplicas exactas del objeto (no debes preocuparte con los tipos predefinidos como **int** porque ya están hechos así). Cuando creemos nuestras clases en futuros números, nos cercioraremos de que así sea y comentaremos este punto con más detalle.

CÓDIGO FUENTE DEL FICHERO DINALIST.HPP

```
     ///////////////////////////////////////////////////
     // Implementación de la HPLS                     //
     // Versión 1.10                                  //
     ///////////////////////////////////////////////////

/**************************************************/
/* Por Javier Basilio Perez Ramas, 1999          */
// High Perfomance List System, Sistema de Lista de Alto Rendimiento
/*
     Esta lista se basa en la idea de una lista enlazada dinámicamente, pero ha logrado optimizar
el tiempo de acceso a los elementos hasta situarlo para muchas aplicaciones en un promedio de O(1)
     La idea es mantener una cache de accesos.
     A la vista de la mayoría de los algoritmos, es fácil darse cuenta de que muchos accesos
seguidos son a posiciones consecutivas de la lista. ¿Porqué no llevar un "cache" de puntero, con la
última posición accedida? El ppo. de localidad nos aconseja hacerlo pues las posiciones cercanas
serán referencias con gran probabilidad.
     No sólo eso, sino que como muchos algoritmos acceden alternativamente a dos puntos del vector
(como mezcla de MergeSort o pivote de QuickSort), podemos acelerar aún más el proceso con una caché
del puntero víctima, es decir, el viejo puntero caché anulado por el último acceso.
     Esta sencilla idea nos ha permitido pasar de un tiempo de ejecución de 23s para 1000 enteros
con inserción a 2.03 s.; el incremento de rendimiento es sencillamente espectacular.
     Además se mantienen punteros al primer y último elemento, así como al central. Esto permite
que, cuando halla un "fallo" de la cache, el tiempo de penalización no sea muy escandaloso.

VERSIONES

 - 1.00 -> PRIMERA VERSION
 - 1.10 -> ALGUNOS BUGS EN INSERTA() FUERON REPARADOS

*/

/* esto activa la cache de accesos. Se puede "jugetear" desactivando esta cte para que no se use el
cache de la HPLS y se comporte como una lista  doblemente enlazada cuasi-tradicial */
#define CACHE // activar la CACHE de accesos

#ifndef LISTAH
 #include "lista.hpp"
#endif

#include <iostream.h>

// Esta macro actualiza la caché de víctima
#define ACTVICTIMA victima=cache; nvictima=ncache

// Esta macro invalida la caché de víctima
#define INVVICTIMA victima=NULL; nvictima=0
```

```
// Implementación de HPLS
 template <class T> class dinalista : public lista<T>
{
      protected:

      struct nodo // estructura de un nodo
            {
                  T contenido; // contenido del nodo
                  nodo *siguiente; // puntero al siguiente nodo
                  nodo *anterior; // puntero al anterior nodo
            };

      typedef nodo* puntero; // puntero a un nodo
      puntero primero,ultimo,central;
      /* En lugar de llevar un puntero al primer elemento de la lista, apuntaremos además al central
y al primero. A la hora de recorrer la lista, se elegirá el punto de partida más cercano. Esto
acelera bastante el acceso en caso de fallo de cache.
       No varía la complejidad del recorrido O(n), pero sí puede mejorar bastante su constante de
proporcionalidad (de hecho nuestra propia experiencia empírica personal muestra una mejora en
algunos casos de casi un 30% de ganancia)
       También llevaremos un puntero al último elemento, para acelerar la inserción */

      puntero cache; // puntero cache
      puntero victima; // cache de victima (el antiguo cache)

      int ncentral; // número del elemento central
      int ncache; // número del elemento cache
      int nvictima; // número de la víctima
      int tamanio; // número de elementos en la lista

      /* Devuelve la dirección y el número del nodo localizado más cercano a uno dado */

      void mascercano(int n,puntero& p,int& ncerca)
      {
            int dprimero,dultimo,dcentral,dcache,dvictima;
            /* ncerca=tamanio;p=ultimo;return;*/
            // calcular las distancias (es fácil)
            dprimero=abs(n-1);
            dultimo=abs(n-tamanio);
            dcentral=abs(n-ncentral);
            dcache=abs(n-ncache);
            dvictima=abs(n-nvictima);

            // si la cache o el victima son invalidos, asegurar que no van a ser elegidos
            if (cache==NULL) dcache=dultimo+1;
            if (victima==NULL) dvictima=dultimo+1;
  #ifdef CACHE
            if (dcache <=dvictima && dcache <= dultimo  &&
                 dcache <= dprimero && dcache<=dcentral)
              // ¿es el nodo cache el más cercano (muy probable)?
             {
                   p=cache;
                   ncerca=ncache;
             }
            else // el nodo cache no es el más cercano
             if (dvictima<=dultimo && dvictima<=dprimero && dvictima<=dultimo)
             {
                   // la víctima es muy cercana, sin embargo
                   p=victima;
```



```
            ncerca=nvictima;
            }
          else // ni el caché ni su víctima valen
#endif
          if (dprimero<=dultimo && dprimero<=dcentral) // el primero es el más cercano
          {
               p=primero;
               ncerca=1;
          }
           else if (dultimo<=dprimero && dultimo<=dcentral) // el último lo es
          {
               p=ultimo;
               ncerca=tamanio;
          }
           else // es el central
          {
               p=central;
               ncerca=ncentral;
          }
     };

     /* Este procedimiento privado devuelve la dirección en memoria del n nodo de la lista */
     puntero ir_a(int n)
     {
          puntero actual;
          int nactual;

          if (n==0) return NULL; // el anterior al primer elemento
#ifdef DEBUG
          if (n<1 || n>tamanio) // ojo al dato la llamadora realmente no sabe qué dice
          {
               cout << "DYNALISTA:: ir_a -> Caller's brain temporaly malfunction";
               cout << endl << "Se intenta acceder al elemento " << n << " de "<<tamanio ;
               exit(999);
           }
#endif
          // Buscar el puntero más cercano
          mascercano(n,actual,nactual); // esto devuelve el puntero inicial más cercano
           // y el número del nodo correspondiente
#ifdef DEBUG
           cout << "Más cercano="<<nactual<<endl;
#endif
          while (nactual != n) // hay que llegar al nodo pedido
                if (nactual<n) { actual=actual->siguiente; ++nactual; } // avanzar
                else { actual=actual->anterior; –nactual; } // retroceder

          // ya estamos posicionados
          return actual; // esto es lo que queríamos
     };

     /* Actualizar el elemento central */
     void actualiza_central(puntero reciente,int n)
     {
          int centro; // nuevo elemento central
          centro=(tamanio/2)+1; // elemento central
          if (centro==n)
          {
               central=reciente;
               ncentral=centro;
```

```
          }
          else
           {
 /* habría un posible error si ir_a usara el viejo puntero central porque ya no es válido. Por eso
provocamos temporalmente que el ncentral sea el último+1, así forzamos a que no sea elegido por ir_a
al ser al menos más lejano  que último*/
           ncentral=tamanio+1;
           central=ir_a(centro);
           ncentral=centro;
           }
       };

     public:

     // Constructor
     dinalista(int t=0)
     {
          T algo;
#ifdef DEBUG
           cout << "Se ha llamado al constructor de DINALISTA."<<endl;
#endif
          ncentral=tamanio=nvictima=ncache=0;
          central=cache=victima=NULL;
          for (int i=1;i<=t;i++)
            aniadir(algo,i);
     };

     // Destructor
     virtual ~dinalista()
     {
          flush(); // vaciar la lista
     };

     // Fuerza la muerte de la lista (en este caso es equivalente al destructor)
     virtual void matar(void)
     {
          flush();
     }

     // Vacia lista
     virtual void flush(void)
     {
          puntero p,sig;
          // recorrer toda la lista eliminándola, si es que hay alguna lista
          if (tamanio>0)
          {
               sig=primero;
               while (sig!=NULL)
               {
                    p=sig;
                    sig=p->siguiente;
                    delete p;
               }
          }
          ncentral=nvictima=ncache=tamanio=0;
          victima=cache=central=primero=ultimo=NULL;
     };

     // Insertar un elemento en una posición de la lista
```

```
      virtual void aniadir(const T& elem,int n)
      {
            puntero eldeantes;
            puntero nuevo;
            int centro;

#ifdef DEBUG
            if (n>tamanio+1) // cuidado
            {
                  cout << "DINALISTA:: Llamada no válida a aniadir() : "<<n;
                  exit(999);
            }
#endif
/* decidir cuál es el camino de exploración más corto, si empezando por el último,por el central o
   por el primero obtener un puntero al elemento anterior */

            eldeantes=ir_a(n-1); // si n=1 eldeantes va a ser NULL, pero no afecta el algoritmo

            /* insertar el elemento */

            /* primero reservar memoria */
            try {
             nuevo=new nodo;
             } catch (...)
            {
                  cout << "FATAL: El sistema operativo ha denegado la concensión de memoria.";
                  exit(999);
            }
            nuevo->contenido=elem; // asignar contenido

            /* enlazar en la cadena de punteros */
            nuevo->anterior=eldeantes;

            /* el único caso especial es el primer elemento, si es el último no importa */
            if (eldeantes != NULL) // no es el primero
            {
                  nuevo->siguiente=eldeantes->siguiente;
                  eldeantes->siguiente=nuevo;
                  // si es el último actualizar el susodicho puntero
                  if (nuevo->siguiente == NULL)
                        ultimo=nuevo;
            }
            else // efectivamente, es el primero
            {
                  if (tamanio>0) // no es vacia
                        nuevo->siguiente=ir_a(1);
                  else // esvacia
                        nuevo->siguiente=NULL;

                   nuevo->anterior=NULL;
                   primero=nuevo; // nuevo primer elemento
            }
/* actualizados los punteros al primero y al último, falta actualizar el nuevo elemento central */

            ++tamanio;
            // actualiza el elemento cache al último acceso
            ACTVICTIMA;
            ncache=n;
            cache=nuevo;
```

```
		actualiza_central(nuevo,n); // se le pasa el nuevo elemento porque si
					    // éste es el nuevo central, no es necesario recorrer
	};

	// Suprimir un elemento de la lista
	virtual void quitar(int n)
	{
		puntero quequito;
#ifdef DEBUG
		if (n<1 || n>tamanio) // por si no es muy listo el chico ...
		{
			cout << "DINALIST::QUITAR -> Programador en estado de embriagüez profunda"
			     << endl;
			exit(999);
		}
#endif
		quequito=ir_a(n); // a ver qué quitamos
		if (quequito->anterior != NULL) // actualizar elemento anterior si existe
		{
			/* actualizar el siguiente del anterior */
			(quequito->anterior)->siguiente=quequito->siguiente;
			if (quequito->siguiente != NULL) // ¿hay siguiente elemento, por cierto?
				/* actualizar el anterior del siguiente */
				(quequito->siguiente)->anterior=quequito->anterior;
			else // es el último de la lista
			{
				ultimo=quequito->anterior;
				quequito->anterior->siguiente=NULL;
			}

			INVVICTIMA;
			cache=quequito->anterior;
			ncache=n-1; // nuevo elemento cache es el anterior
		}
		else // no hay anterior
		if (quequito->siguiente != NULL) // sin anterior pero con siguiente
		{
			// el anterior del siguiente no es nada
			(quequito->siguiente)->anterior=NULL;
			primero=quequito->siguiente; // por cierto, es el nuevo PRIMERO
			// el siguiente será el nuevo elemento caché
			INVVICTIMA;
			cache=quequito->siguiente;
			ncache=n+1;
		}
		else // el que quitamos era el único nodo de la lista
		{
			// esta comprobación es algo absurda, pero nunca se sabe
			if (tamanio!=1)
			{
				cout << "Algo realmente horrible ha sucedido al quitar UN elemento.";
				exit(999);
			}
			// ahora la lista está vacia
			primero=ultimo=central=cache=victima=NULL;
			ncentral=ncache=nvictima=0;
		}
```



```
        --tamanio; // es lógico, si quitamos un elemento tenemos uno menos
        // ahora hay que actualizar el elemento central ...
        if (primero!=NULL) // sólo si hay algo
            actualiza_central(NULL,-1);      // los parámetros hacen que sea imposible
                                // tomar este elemento como referencia (VER CENTRAL)
        // resta deshacerse del nodo viejo
        delete(quequito);
    };

    // Acceder a un elemento de la lista
    virtual void leer(T& que,int n)
    {
        if (n>tamanio)
        {
            cout<<"DINALIST:Lectura invalida"<<endl;
            exit(1);
        }
        puntero p;
        p=ir_a(n);
        que=p->contenido;
        // actualizar cache
        ACTVICTIMA;
        cache=p;
        ncache=n;
    };

    // Devuelve un puntero a un elemento de la lista
    virtual T* leer_referencia(int n)
    {
        T *punterico;
        if (n>tamanio)
        {
            cout<<"DINALIST:Lectura invalida"<<endl;
            exit(1);
        }
        puntero p;
        p=ir_a(n);
        punterico=&(p->contenido); // eso es lo que buscamos
        // actualizar cache
        ACTVICTIMA;
        cache=p;
        ncache=n;
        return punterico;
    }

    // Sobreescribir un elemento existente de la lista
    virtual void asignar(const T& elem,int n)
    {
        puntero p;

#ifdef DEBUG
        if (n<1 || n>tamanio)
        {
            cout << "DINALIST::ASIGNAR error en llamada con parámetro "<<n;
            exit(999);
        }
#endif
        p=ir_a(n); // encontrar el nodo n
        p->contenido=elem;
```

```
            // actualizar el cache de acceso
            ACTVICTIMA;
            cache=p;
            pcache=n;
        };

        // Intercambiar dos elementos de la lista
        virtual void intercambiar(int na, int nb)
        {
            puntero anta,siga,a;
            puntero antb,sigb,b;
            int temp;
            if (na==nb) return; // uno puede esperarse cualquier burrada del programador ...

            if (na>nb)   // asegura que na es el más pequeño
            {
                temp=na;
                na=nb;
                nb=temp;
            }

            // localizar los nodos a intercambiar en la lista enlazada
            a=ir_a(na);
            b=ir_a(nb);

            // Ahora ver sus conexiones
            anta=a->anterior;
            antb=b->anterior;
            sigb=b->siguiente;
            siga=a->siguiente;

            // si vamos a intercambiar el nodo central, mejor lo corremos una posición
            // ¡por no hacer esto la lista fallaba y nos llevó más de 2 horas encontrarlo!
            if (ncentral==na || ncentral==nb)
            {
                ncentral=na-1;
                central=anta;
            }

            // Vamos a ver el caso de que 'b' sea adyacente a 'a' (el siguiente)
            if (siga==b) // caso especial,b justo después de a
            {
#ifdef DEBUG
                if (na!=nb-1)
                {
                    cout << "Joooorrrrrrl: en dinalista::intercambio"; exit(999);
                }
#endif

                // ¡A ANTES QUE B! Hay que poner B PRIMERO, y después A poner B primero
                b->anterior  = anta; // el anterior de b es el antiguo anterior de a
                b->siguiente = a;    // el siguiente de b es ahora a

                // poner A después
                a->siguiente = sigb; // el siguiente de a es el que seguía a b
                a->anterior  = b;

                //  el nodo n-2 debe apuntar ahora a B en su siguiente
```



```
                if (anta!=NULL) // sólo si existe DICHO nodo
                     anta -> siguiente= b;
                else primero=b; // en ese caso, B es el nuevo primero

                // el nodo n+1 debe apuntar ahora a A en su anterior
                if (sigb!=NULL) // si existe ese nodo
                     sigb -> anterior = a; // ahora a es el nodo n
                else ultimo=a; // a es el nuevo último nodo
                }
           else // a y b bien separados
           {
                // sabemos que a está antes que b así que sabemos que ni a es el último, ni b es
                // el primero (esto nos va a ahorrar unas poquitas comparaciones extra)

                // Colocar b en el lugar de a
                b->anterior=anta;
                if (anta != NULL) // a no era el primero
                     (anta->siguiente)=b;
                else // anda, pues sí que lo era
                     primero=b;
                // actualizar el siguiente
                b->siguiente=siga;
                 siga->anterior=b;

                // Colocar a en el lugar de b
                a->anterior=antb;
                antb->siguiente=a;

                // actualizar el siguiente
                a->siguiente=sigb;

                if (sigb != NULL) // b no era el último
                     sigb->anterior=a;
                else // pues sí
                     ultimo=a;
           }
           // tomaremos el primer elemento del intercambio como nuevo cache y al otro como víctima
           victima=a;
           nvictima=nb;
           ncache=na;
           cache=b;
      };

       // Este operador sobrecargado equivale a “leer”
      virtual T operator[](int n)
      {
           puntero p=ir_a(n);
           // actualizar cache con la lectura
           ACTVICTIMA;
           cache=p;
           ncache=n;
           return p->contenido;
      };

      // devolver el numero de elementos en la lista
      virtual int cuantos(void)
      {
           return tamanio;
      }
};
```

PROGRAMACIÓN AVANZADA EN DAP

Por: Javier Basílio Pérez

1. COMO MANEJAR OBJETOS CONSUMIBLES


omenzamos aquí una sección dedicada a la programación avanzada en el parser DAP. Estos artículos no pretenden ser una descripción de cómo se usa el parser o de cómo hacer una aventura con él: para eso ya está el manual de DAP, donde esto se explica detalladamente con todo tipo de ejemplos. Nosotros nos dirigiremos al usuario que ya sabe usar DAP y desea profundizar en él, explicando técnicas para enriquecer tus aventuras. DAP es un parser que busca la sencillez absoluta y por eso hay algunas características avanzadas que no se encuentran explícitamente en él; esta serie te ayudará a implementar estas características haciendo un hábil uso del microlenguaje de orgones.

En esta primera entrega trataremos del manejo de objetos consumibles; es decir, que se van gastando según los vamos usando. Hay varios enfoques para tratar este problema; veremos uno de los muchos posibles.

Desarrollaremos la técnica mediante un clásico ejemplo; el de la linterna que se va consumiendo según la vamos usando. La linterna tiene una cierta carga eléctrica que va bajando según permanece encendida. Hay ciertas localidades que están a oscuras, en las que no podemos ver nada sin la linterna.

Para mantener la carga de la linterna, vamos a hacer uso del flag auxiliar del que disponen todos los objetos a partir de la versión 4 de DAP. En DAP, los objetos están definidos por 4 campos:

- Un número de objeto que lo identifica internamente
- Una descripción de texto
- Un precio
- Un flag numérico auxiliar que no es usado por DAP y queda libre para el usuario.

Nosotros haremos uso precisamente de este flag: en él almacenaremos el estado del objeto consumible.

Vamos a imaginar que tenemos una aventura con los siguientes elementos:

- Una localidad (nº 73) a oscuras (si tenemos más localidades, el proceso es análogo)
- Un objeto linterna con número 1
- Un objeto pilas con número 2

Al principio, la linterna está descargada y su flag auxiliar vale cero. La linterna además podrá estar encendida o apagada; esto lo controlaremos mediante uno de los flags globales del programa, por ejemplo, el flag 10:

Si vale 1, la linterna está encendida
Si vale 0, la linterna está apagada.

Además, para las comparaciones vamos a suponer que el flag 98 siempre vale 0 (es el valor que tiene inicialmente y no cambiará sino lo tocamos)

Así, por ejemplo, es muy sencillo crear una rutina para que el jugador pueda examinar la linterna y recibir información sobre si está encendida o apagada, y la carga que le queda:

Orden: ex linterna

Rutina EDAP:

```
OBJ 1,M1 // comprueba si el usuario tiene la linterna
PRINT M2 // escribe la descripción de la linterna
PRINT M3 // escribe la descripción de la carga de la linterna
OBFTOVAL 1 @2 // copia el flag de estado de la linterna al flag 2
PRINTVAL @2 // imprime el contenido del flag 2
VAL @10 1 M4 // comprueba si la linterna está apagada (si es así, imprime 4 y cancela)
PRINT M5 // linterna encendida
RET
```

Donde los mensajes valen, por ejemplo:

```
1-  "No tienes ninguna linterna. Cómprate gafas"
2-  "La linterna es compacta y liviana"
3-  "La linterna tiene la siguiente carga:"
4-  "La linterna está apagada."
5-  "La linterna está encendida."
```

También supondremos que tenemos un m
ensaje 299 vacío, para los siguientes procedimientos en los que halla orgones de comprobación y no deseemos imprimir nada.

A la vista de esto, es muy fácil crear los procedimientos para encender y apagar la linterna, por ejemplo, este el que usaríamos para encenderla:

Orden: encender linterna (posible sinónimo: enciende linterna)

Rutina EDAP:

```
OBJ 1,M1
CAMVAL @10,1,M5
RET
```

De acuerdo. Ahora bien, ¿cómo hacemos que la linterna se gaste cuando está encendida?. Es preciso crear un procedimiento global (un glolo, en la terminología de DAP) que se ejecute en cada ciclo de juego. Nada más sencillo; sólo hay que crear un procedimiento glolo asociado a la localidad cero (los glolos son procedimientos que se ejecutan en una cierta localidad; si los asociamos a la localidad 0, hacemos que se ejecuten en todas las localidades). Por cierto, observa que no comprobamos que el jugador tenga el objeto linterna. La utilidad de esto es que la linterna se va consumiendo también si la dejamos si está encendida, como es lógico.

Glolo para la localidad 0

```
VAL @10,1,M299 // si no está encendida retorna (imprime mensaje vacio)
OBFTOVAL 1,@2 // copia el flag de estado de la linterna al flag 2
DEC @2 // decrementa el flag 1 (la linterna consume energía)
VALTOOBF @2,1 // copia el valor decrementado de nuevo al flag de estado del objeto
VAL @2,0,M299 // si la linterna no se gastó (flag 2=0) retorna con mensaje vacio
OBJG 1,9 // si tiene la linterna, salta a la línea 9 (hay que informar de que se apaga)
CAMVAL @10,0,M299 // apaga la linterna, pero no informa porque no se tiene el objeto
RET // termina
CAMVAL @10,0,M6 // apaga la linterna e informa porque el jugador sí tiene el objeto
RET // termina
```

(el mensaje 6 es algo así como "La linterna se ha quedado sin energía")

El siguiente asunto a resolver es la carga de la linterna. Pero nada más sencillo. Por ejemplo, supongamos que el jugador tiene una pila, el objeto número 2 (quizás la halla obtenido de una máquina recarga-pilas en medio de una caverna repleta de tesoros).

Orden: carg linterna (posible sinónimo: met pila)

Procedimiento EDAP:

```
OBJ 1,M1
OBJ 2,M7
OBFTOVAL 1,@2
ADDVAL @2,1000 // le ponemos 1000 más de carga, por ejemplo
VALTOOBJ @2,1
LOBJ 2,999,M8 // ¡la pila ya no está disponible (la movemos a la localidad fantasma)!
RET

Mensaje 7: "No tienes la pila."
Mensaje 8: "Has cargado la linterna usando la pila"
```



Éste es un método algo rudimentario, por ejemplo, la pila se "evapora" después de cargar la linterna y no puede ser recuperada, ni siquiera vacía, pero como primera aproximación es válida. En artículos siguientes veremos como hacer objetos contenedores en DAP.

Tan sólo resta ver qué hacemos con la localidad a oscuras. Pero nada más sencillo; sólo hay que crear un glolo en la localidad que deseemos (73 en nuestro ejemplo) que varíe la descripción de la misma según la linterna esté encendida o no.

Glolo para localidad 73

```
CAMTXT 73,M10 // pone el texto de descripción oscura
VAL @10,1,M12 // retorna si la linterna no está encendida
CAMTXT 73,M11 // la linterna estaba encendida, rectifica al texto de descripción normal
RET

Mensaje 10: "No logras ver nada"
Mensaje 11: "Estás en el recolector de tesoros" (descripción habitual de la localidad 73)
Mensaje 12: "Te sumes en las más profunda oscuridad..."
```

Y con esto hemos acabado esta primera entrega, que espero que le halla sido útil a alguien. Tan sólo recordaros la página WEB del DAP, donde podéis encontrar la última versión del mismo en línea junto a todo tipo de documentación:

http://www.geocities.com/sytovos/dap.htm

ISAAC ASIMOV
MEMORIAS

Por: Francisco José Súñer

Título Original: I, Asimov
Año de Publicación: 1994
Editorial: Ediciones B
Colección: Vib nº 138/5
Traducción: Teresa de León
Edición: Febrero de 1998
ISBN: 84-406-8120-8
Precio: 1.100 pts. (6,61 eur.)

saac Asimov, sin ningún género de dudas, es el autor más conocido, respetado y reverenciado de la ciencia-ficción.

Hay otros muchos autores que le superan en todo o en parte, como literatos, como fabuladores, como articulistas, pero en lo que ninguno puede superarle es en popularidad y, cosa casi imposible, en el número obras publicadas.

En estas son las segundas memorias que escribió Asimov, las primeras, en 1977, se habían agotado hacía mucho y toda una serie de cuestiones crematísticas y temporales (el propio Asimov era consciente de que se le acababa el tiempo) le llevaron a finales de los ochenta a escribirlas de nuevo, pero de una forma muy distinta a aquella primera, que se trataba simplemente de una exposición cronológica de su vida y obra.

En esta ocasión, y a instancias de Janet, su segunda mujer, Asimov no se limitó a desgranar los años. Escribió cortas entradas independientes de todo aquello que de alguna u otra forma le había marcado durante su vida, para finalmente ordenarlas cronológicamente. Por eso, en algunas ocasiones se producen extraños saltos en el tiempo e incluso repeticiones de sucesos, que únicamente hacen confiar en la sinceridad de Asimov al redactar este libro.

Realmente hay pocas cosas que un seguidor de la obra de Asimov (y prácticamente cualquier aficionado a la ciencia-ficción) no supiera ya. En sus recopilaciones (COMPRE JÚPITER, LA EDAD DE ORO DE LA CIENCIA-FICCIÓN, Etc.) Asimov ya situaba de una forma notablemente desenfadada el tiempo y las circunstancias que rodeaban la escritura del mismo.

Sin embargo, el tono que usa en estas MEMORIAS no es nada festivo, y aunque ameno, como siempre en Asimov, tampoco es indulgente consigo mismo y así, confiesa que cuestiones, como su doctorado en biología, que casi todo el mundo ha usado como demostración palpable de que la ciencia-ficción no es un género escrito por patanes fantasiosos, fue el resultado final de una carrera académica desastrosa y eso si, muy sacrificada.

Tampoco le duelen prendas al reconocer su limitada capacidad como escritor. El era el primero en saber, y muy bien, cuales eran sus capacidades, que podía y que no



podía escribir y como podía hacerlo. En este sentido es injusto acusar a Asimov de escritor poco dotado. Tanto él como sus compañeros de generación (Pohl, Kornbluth, Heinlein, Simak, Williamson...) se iniciaron en los pulp. Escribían para un medio en el que era necesario entretener a un lector sin grandes exigencias, únicamente la de rentabilizar al máximo los diez centavos que pagaba por el tocho de papel, si este objetivo se conseguía con un mínimo de dignidad, el trabajo estaba asegurado, y desde luego hay que tener en cuenta que Asimov no es precisamente un autor olvidado, y que su calidad intrínseca le hizo transcender a aquella primera época oscura, en la que la mediocridad era moneda común.

Asimov era muy consciente de esta y de otras muchas de sus limitaciones. Quien pensara que era dueño de un ego de dimensiones galácticas y se creía un genio sin parangón está muy equivocado. En estas memorias describe paso a paso como él mismo descubrió que de genio, nada. Académicamente, de párvulo casi prodigio paso a bachiller del montón para convertirse finalmente en un universitario bastante malo. Sin embargo había algo que sabía hacer mejor que la mayoría de sus condiscípulos, y eso era escribir, y con el tiempo se dio cuenta de que era capaz de hacer con una soltura y un dominio de la situación poco comunes algo que a la gente le aterra; hablar en público.

Con su familia tampoco es especialmente indulgente. A su hijo le califica sin reparos como vago inútil, admite sin ambages el tremendo error que fue su primer matrimonio, el gran respeto que sentía por sus padres, la adoración por su hija y su segunda mujer y, las diferentes relaciones con sus hermanos. En este sentido tampoco se muestra conciliador a la hora de hablar de sus colegas de profesión. Frío, distante y casi crítico, aunque no por ello menos admirador de su obra, con Heinlein, cálido con sus amigos, como Frederik Pohl o Lester del Rey, y profundamente agradecido a la mayoría de sus editores, entre los que Campbell siempre ocupó para él el número uno.

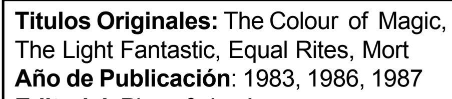
Este libro aclara también una leyenda negra que siempre ha rodeado a Asimov; la existencia o no de negros que le ayudaran a escribir la ingente cantidad de obras publicadas con su nombre en la portada. El tema es complejo por cuanto es el propio Asimov quien no se atribuye algunas de las obras que firma. Cuando ya era muy famoso, y quizá más como divulgador científico que como autor de ciencia-ficción, era continuamente requerido para prestar su nombre a obras de terceros. El objetivo obvio de todos (terceros y editores) era aprovechar el tirón comercial del apellido Asimov, pero era el propio Asimov, quien con una honestidad que le honra, insistía en participar de algún modo en el libro, ya fuera haciendo correcciones, aportando datos o escribiendo entradillas. Desde luego Asimov no era tonto, y si no le era posible colaborar, no tenía inconveniente en prestar su nombre si la obra le satisfacía, pero no la anotaba en su registro personal, e incluso insistía en que el verdadero autor fuera en primer lugar en la portada. Noble, pero generalmente, ignorada petición.

Otra cuestión que tira por tierra la teoría de los negros es que Asimov era adicto a la escritura. Para él, escribir era su mejor terapia, su mejor entretenimiento y su mejor droga. Podía pasarse horas escribiendo ajeno a todo lo que le rodeaba, y en las ocasiones en las que la vida le castigaba con sus peores reveses, se refugiaba en la escritura para evadirse de la realidad. No es extraña, por tanto, su fabulosa y extensa producción.

En fin, este es un libro interesante en todos los aspectos, entretenido (Asimov podría no ser un genio, pero le encantaba hablar de si mismo) y sobre todo con un enorme aporte de datos que harán comprender mejor la obra y motivaciones de este autor. ■

TERRY PRATCHETT

LA SAGA DEL MUNDODISCO

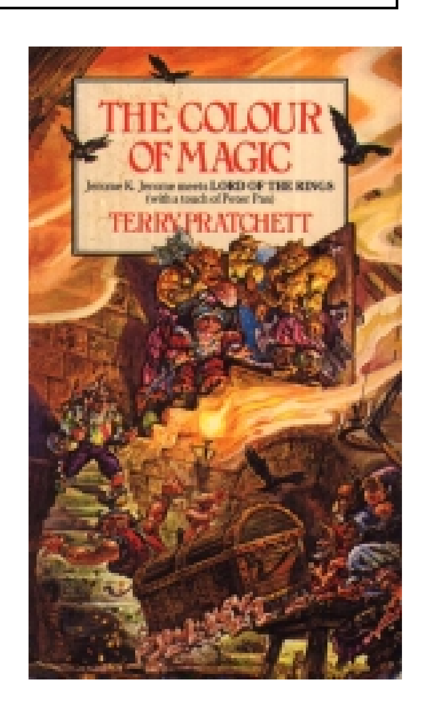

Por: Francisco José Súñer

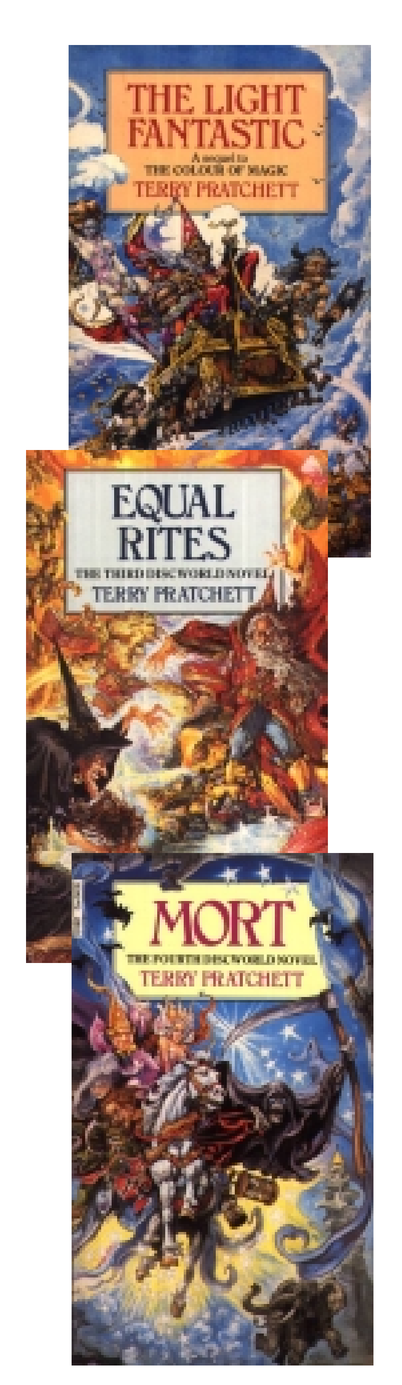
Titulos Originales: The Colour of Magic, The Light Fantastic, Equal Rites, Mort
Año de Publicación: 1983, 1986, 1987
Editorial: Plaza & Janés
Colección: Jet 342/1-4
Traducción: Cristina Macía
Edición: Enero de 1998
ISBN: 84-7386-416-6, 84-01-47944-4
.... 84-01-47942-8, 84-01-47941-X


n primer lugar Tengo que reconocer que inicié la lectura de esta saga (¡Inter-minable Saga! ¡Prattchet ha escrito ya un mínimo de 23 volúmenesi) justo tras acabar LA SAGA DE LA TIERRA MORI-BUNDA, y lo cierto es que la diferencia que hay entre una y otra es abismal. No estoy diciendo con esto que LA SAGA DEL MUNDODISCO sea mala, simplemente que LA SAGA DE LA TIERRA MORIBUNDA es tan buena que la comparación no es posible. Algún fanático de Pratchett dirá que no tengo ni idea de lo que estoy hablando. Posiblemente. Pero si se, y muy bien, lo que pienso, y lo que pienso es lo que está escrito una miaja más arriba.

Advertido esto, no me queda más remedio que decir que me he partido el pecho a reír con las aventuras de los habitantes del Mundodisco, con las ocurrencias de Pratchett y con la parsimonia de A'Tuin.

Aunque todo en su justa medida. Las aventuras y peripecias de Ricenwind y compañía no deja de ser en bastantes casos previsible y las gracias y gracietas de Dos Flores y Cohen poco originales. Divierten, eso si, pero la sensación general es la de haber visto o leído antes esos mismos efectos cómicos. Quizá son lo únicos efectos cómicos posibles y si resultan tan evidentes es porque Pratchett los utiliza con descaro y cargando las tintas.





Otra cosa son las ocurrencias de Pratchett. El mismo concepto de mundo transportado está tomado de alguna de las millones de formas de explicar la creación que se ha inventado la humanidad, pero hasta que Pratchett no lo hizo a nadie se le ocurrió que esa gran tortuga que navega en el río del Universo tuviera sus propias ideas, (laaargas ideas) o plantear la importancia de saber si es macho o hembra. Cuestión muy fundamental por cuando de eso (y de ciertas urgencias muy naturales) depende la llegada del Fin del Mundo.

Y lo mejor de todo es la Magia que inunda el Mundodisco. Una magia con color y una personalidad muy marcada, tanto que los hechizos tienen vida propia y sólo los grandes magos son capaces de dominar y memorizar.

De entre estos cuatro primeros volúmenes editados por Plaza & Janés los dos primeros me han parecido los más flojos. Divertidos, livianos, de lectura agradable pero con esa sensación de la que hablaba antes de estar ya muy visto. LA LUZ FANTÁSTICA aventaja quizá a EL COLOR DE LA MAGIA en la ácida parodia a Conan el Barbaro, personalizado en un Cohen tan decrépito como peligroso, mejor conseguida que la del turista occidental que todo lo sabe encarnada por Dos Flores, .

Con RITOS IGUALES y MORT la cosa mejora. Ya no son las desgracias de Rincewind y la ingenua intrepidez de Dos Flores las que guían los relatos. Perdidos de vista Dos Flores y Rincewind y al no tener tanto afán paródico como las dos primeras, estas novelas no tiende al humor desaforado, y por eso me resultaron de lectura mucho más agradable.

Saga de lectura refrescante y divertida, con la única pretensión de hacer reír al lector. Completamente recomendable. ¡Esperemos que Plaza & Janes complete la publicación de todos los volúmenes escritos ya por Pratchett! ■

FEEDBACK

Muchos de vosotros lloraréis, os desesperaréis, rasgaréis vuestras vestiduras o pasaréis olímpicamente de este párrafo, pues siempre lo dedico a lo mismo, número tras número, CAAD tras CAAD...

¿Para qué sirve el Feedback? Pues para que el CAAD sea cada día más del gusto de sus socios, que son los que mandan. El proceso es simple: Basta con responder en una tarjeta postal las cuestiones que se os formulan, y además participáis en el sorteo de una suscripción gratuita al CAAD entre todas las tarjetas recibidas.

Precisamente el último agraciado ha sido Pedro Pinillas Marqués, socio 562, que recibirá la 13ª Suscripción CAADiana en su casa sin necesidad de aflojar ni un ochavo... salvo lo que le costó tarjeta y sello, claro. ¿Te parece una buena inversión? Suscripción gratuita a un fanzine amoldado a tus preferencias, mmm...

RESULTADOS DEL CAAD 32

1.- Afortunadamente no ha habido ninguna indicación estrambótica de puntuación esta vez... ¡al menos en comparación a lo que nos estábamos acostumbrando! Hemos alcanzado un 10 como puntuación más alta, mientras que un 7,5 fue la más baja y la más precisa 9,7682832. La media ha quedado establecida en 8,95.

2.- Ningún socio pensó que el CAAD 32 era peor que el 31. Un 18% los equiparó, y el resto, abrumador 82% lo consideró mejor.

3.- Hay mucha diversidad entre lo que más ha gustado, pues se reparte entre variadas secciones, como Noticias, los Informes... ¡incluso el que haya salido!

4.- Lo que menos ha gustado es el corto número de páginas, y la no inclusión de más cursos e información sobre parsers.

5.- Por último, absolutamente todos los socios votantes están interesados en acceder a Internet en un plazo más o menos breve. Desde aquí os aseguramos que será toda una experiencia.

CUESTIONES DEL CAAD 34

1.- ¿Cómo calificarías este número? (0 = horrible, 10 = perfecto)
2.- ¿Es mejor o peor que el número anterior?
3.- ¿Qué te ha gustado más?
4.- ¿Y menos?
5.- ¿Qué podemos hacer para resucitar a la aventura conversacional en España?